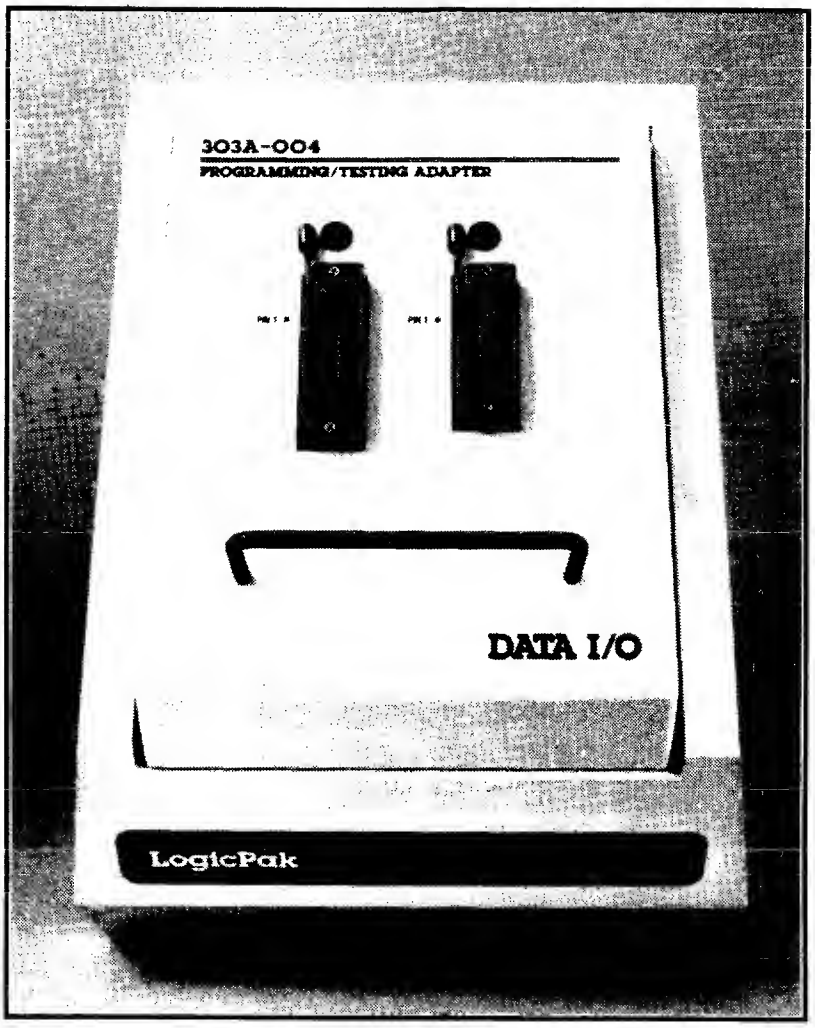



# *LogicPak™*

**OCT 83 REV D** **10-950-1942**
**Applies to configuration 950-1942-004**

*Data I/O has made every attempt to ensure that the information in this document is accurate and complete. However, Data I/O assumes no liability for errors, or for any damages that result from use of this document or the equipment that it accompanies.*

*Data I/O reserves the right to make changes to this document without notice at any time.*

*NOTE*

*Before using the LogicPak™, read section 1 for programmer mainframe compatibility information.*
*If you are not familiar with programmable logic device technology, it may be helpful for you to read the glossary in appendix B to familiarize yourself with the terminology before reading any of the system documentation.*

# TABLE OF CONTENTS

SECTION 1. INTRODUCTION



SECTION 2. INSTALLATION



SECTION 3. OPERATION

# LIST OF FIGURES

# LIST OF TABLES

# SECTION 1

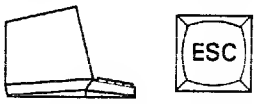
# INTRODUCTION

## 1.1 OVERVIEW

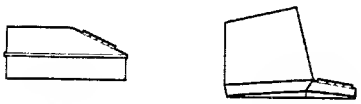
Data I/O's Programmable Logic Development System (PLDS) provides data development, programming, and testing support for logic device families from most semiconductor manufacturers. Once the programmer data RAM (random access memory) has been loaded with a specific fuse pattern, devices can be programmed by installing the appropriate adapter in the PLDS. With the PLDS, system designers can now use one integrated system to specify logic device functions using Boolean equations or truth tables, can program, and can functionally test devices.

This manual describes the components and operation of the PLDS. Subjects addressed in this manual and the corresponding sections are listed in table 1-1. Use this list as a quick reference point of the major sections in this manual.

Throughout this manual, the entries that you are to make from either the programmer or terminal are indicated by the entry enclosed in a key symbol. For example,




indicates the "ESCAPE" key on the terminal keyboard should be pressed. In addition, the symbols shown below are used to indicate modes of operation and prompts.



29A TERMINAL

Table 1-1. Using the LogicPak™ Manual

| Subject | Section |
|---|---|
| Interrelationships of system components | 1.2 |
| Functions of system components | 1.2 |
| Conceptual overview of data development | 1.4 |
| Conceptual overview of programming | 1.4 |
| Conceptual overview of testing operations | 1.4 |
| Installation procedures for LogicPak™ and adapter installation | 2.0 |
| Basic operation instructions for LogicPak™ with the 29A Universal Programmer | 3.0 |
| Exact key sequences for operating LogicPak™ with the 29A Universal Programmer (refer to your programmer manual for System 19 and 100A key sequences) | 3.0 |
| Maintenance for LogicPak™ | 4.0 |
| Calibration for LogicPak™ | 4.0 |
| Troubleshooting for adapters (also in each appropriate adapter manual) | 4.0 |
| Circuit descriptions for LogicPak™ (in appropriate adapter manual for the adapters) | 5.0 |
| Block diagram for LogicPak™ (in appropriate adapter manual for the adapters) | 5.0 |
| Schematics for LogicPak™ (in appropriate adapter manual for the adapters) | 5.0 |
| Transfer of information between data preparation system and a device programmer | Appendix A |
| Glossary | Appendix B |
| Logic schematics | Appendix B |
| Data I/O service centers | Back of manual |
| Warranty information | Back of manual |

™ LogicPak is a trademark of Data I/O.

## 1.2 PLDS SYSTEM OVERVIEW

The PLDS provides a universal means for developing data, programming, and testing a variety of logic devices in engineering and production environments. This modular system comprises the LogicPak™, a programmer, adapters, and a terminal (see figure 1-1).

Like the Data I/O programming paks, the LogicPak™ is designed to interface with a standard programmer such as the 29A Universal Programmer (see section 1.3 for programmer compatibility). The LogicPak™ contains the firmware and electronics necessary for PLDS operations and controls the flow of data to and from the user as well as throughout the system. In this manual we will refer to the operational procedures for the 29A Universal Programmer; refer to your programmer manual for System 19 and 100A production programmer key sequences.

Software tables within the LogicPak™ store values for programming variables, including pinouts, voltage levels, and timing. This allows high-speed operation with minimum firmware.

The PLDS has two types of adapters: programming/testing (P/T) adapters and design adapters. There is one P/T adapter for each manufacturer's device family. These adapters fit onto the top of the LogicPak™ and provide the supplementary, device-specific software to convert the LogicPak™, programmer and a terminal into a programmable logic development system. A list of available logic devices and the hardware necessary for data development, programming, and testing of each device is included with each adapter manual and is on the "wall chart" (Data I/O Comparison Chart of Programmable Devices).

As Data I/O increases the capabilities of the LogicPak™ to program new or additional devices, firmware updates will be available for existing adapters to add new devices to existing device families. New adapters may also be added to the PLDS to accommodate new device families.

The PLDS programs a wide variety of logic devices, such as FPLA (field programmable logic array), PAL® (programmable array logic), FPLS (field programmable logic

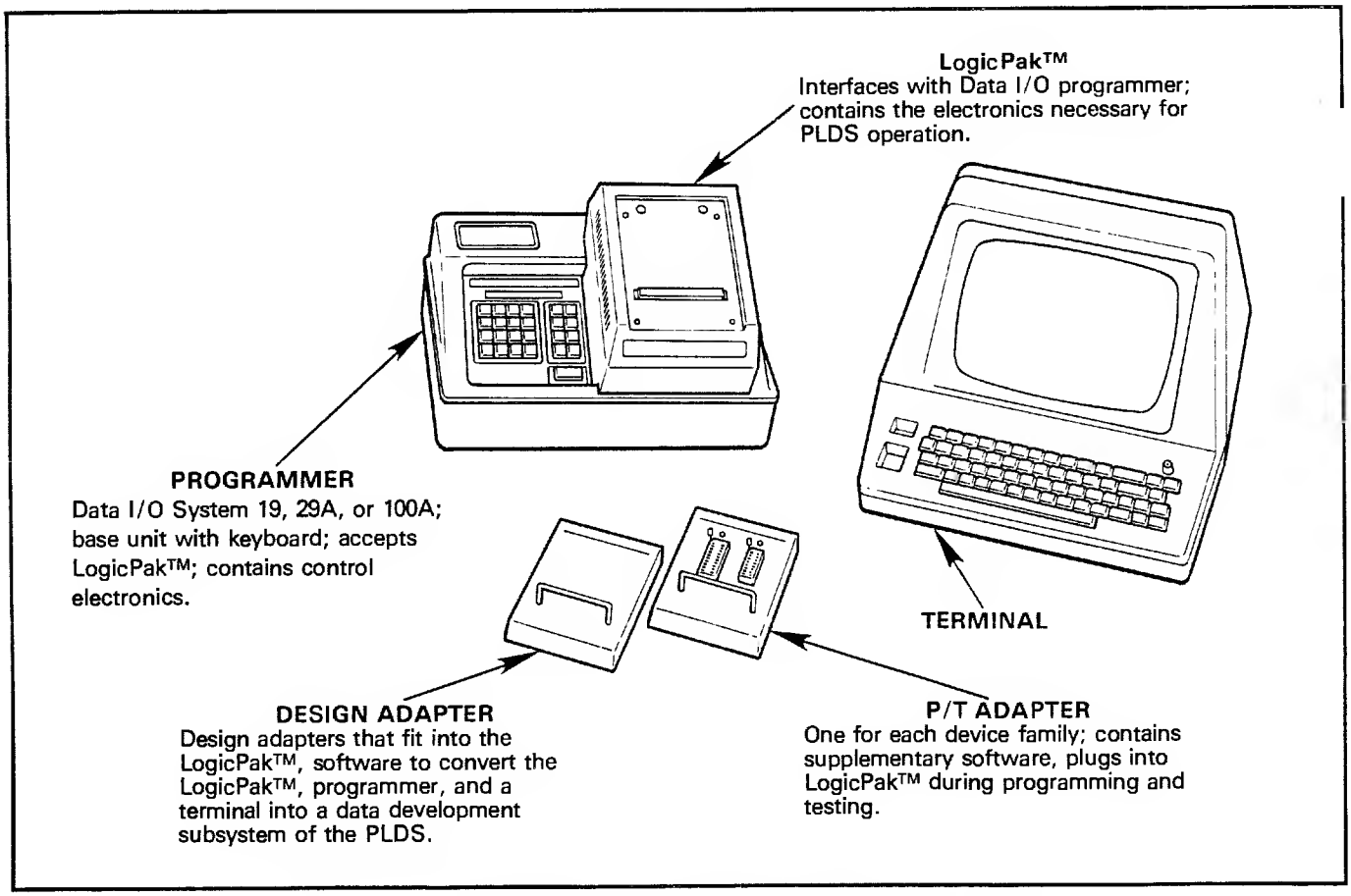



**Figure 1-1. Programmable Logic Development System Components**

® PAL is a registered trademark of Monolithic Memories, Inc.

sequencer), and FPGA (field programmable gate array), with a minimum of equipment changes. Because P/T adapters configure the LogicPak™ to specific manufacturers' device families, the only hardware you have to change when you change device families is the P/T adapter.

Programmed devices can be functionally tested using an optional method developed by Data I/O called the Logic Fingerprint™ test. This unique new test, which can be used for combinational and sequential devices, finds defective devices that could pass a routine fuse verify operation. The Logic Fingerprint™ test is fast and easy and does not require you to generate test vectors. See section 1.4.3 for a full description of this feature.

A structured test can also be performed on programmed devices, based on the test vectors you generate. This additional testing usually will not be necessary with most combinational devices because the Logic Fingerprint™ tests them adequately. With some sequential devices, the Logic Fingerprint™ test cannot test all active states. It checks a statistically significant sample, but not necessarily a specific state you desire. When it is necessary to define certain test sequences explicitly, the structured test allows you to write test vectors and test specific states for full assurance of proper functionality.

These test vectors are applied to the device in sequence and apply desired inputs and check for correct outputs. When used in conjunction with the Logic Fingerprint™ test, they greatly increase the likelihood of finding defective devices. When entered, structured tests are always performed prior to the Logic Fingerprint™ test. Device power remains applied between structured testing and Logic Fingerprint™ testing, therefore allowing initialization of sequential devices. Some sequential devices require structured test vectors and/or selection of the Logic Fingerprint™ test starting vector to avoid false errors when performing the Logic Fingerprint™ test. These requirements are device and fuse-pattern dependant; refer to the appropriate P/T adapter manual for specific examples of the devices that are subject to special considerations.

New data development features simplify design efforts. Monolithic Memories' PALASM (PAL assembler) and Signetics H&L (high-and-low) logic development software are contained in the design adapters.

PALASM is a development language that allows you to describe the desired operation of a PAL device using Boolean equations. Signetics H&L is used to design IFL (integrated fuse logic) devices using logic function tables.

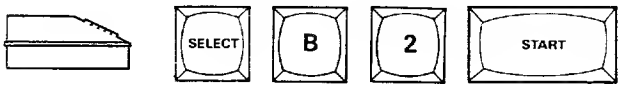
## 1.3 PROGRAMMER COMPATIBILITY

In addition to the 29A, Data I/O's System 19 and 100A Production Programmers are compatible with the LogicPak™. To be compatible with the LogicPak™, your programmer may require a hardware and/or firmware update, depending on the model, configuration, and age. The following information will help you determine if your programmer requires updating. If you find that your programmer does require updating, contact your nearest Data I/O service center. (A list of service centers is at the back of this manual.)

- System 17—The System 17 must be converted into a System 19 with the latest firmware installed and with the latest hardware modifications.
- System 19—Check to determine if your System 19 contains a 702-1520 or 702-1980 controller board by performing the following steps:

  1. Remove the programming pak.
  2. Remove the metal shield (if any) (see section 2.2, figure 2-1).
  3. Count the number of EPROM firmware sockets located just behind the pak interface connector. If there are four sockets, it is a 702-1520 board. If there are eight sockets, it is a 702-1980 board.

  If your System 19 contains a 702-1520 controller board, check the modification status sticker (figure 1-2) on the bottom of the programmer. If the sticker is not there, or if the "1" is marked, your System 19 requires hardware and firmware updating. Contact the nearest Data I/O service center. If "2" is marked, your System 19 is compatible with the LogicPak™. If your System 19 contains a 702-1980 controller board, it may require a firmware update. To display the configuration number of the firmware in your programmer, press




If the configuration number displayed is "3599" or "CC8B" your firmware needs updating.

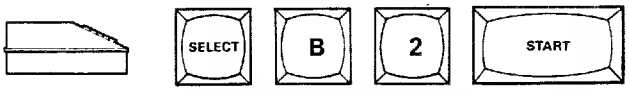
MODIFICATION STATUS

| 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|
| ● | 0 | 0 | 0 | 0 | 0 |

**Figure 1-2. Example of a Modification Status Sticker**

™Logic Fingerprint test is a trademark of Data I/O.

- 29A Universal Programmer—To be compatible with the LogicPak™, the 29A programmers must have Rev (revision) C or later firmware. To determine the configuration of the firmware in your 29A, press

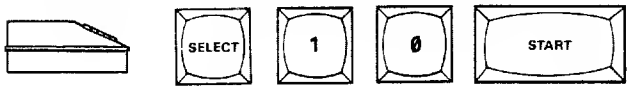



and observe the display. If the hexadecimal number matches one listed in table 1-2, your firmware needs to be updated.

Table 1-2. Programmer Compatibility

| Model | Revision | Configuration Number |
|---|---|---|
| 29A | A<br>B | 1ECA<br>20A4 |
| 29A (with computer remote control) | A<br>B | BB41<br>C00B |
| 100A | A<br>B<br>C<br>D | 917F<br>9405<br>9DEE<br>9BED |

- 100A Production Programmer—To be compatible with the LogicPak™, the 100A programmers must have Rev E or later firmware. To determine the configuration of the firmware in your 100A, press




and observe the display. If the hexadecimal number display matches one listed in table 1-2, your firmware needs to be updated.

## 1.4 THEORY OF OPERATION

Logic devices are programmed by first defining the logic structure of a programmable logic device. This first step is known as data development. The logic structure is specified using a function table or a set of Boolean equations. These Boolean or function table designs must be translated into a fuse pattern in RAM using a design adapter.

Once the fuse pattern is translated, you can program the device using a P/T adapter. The programmer can then blow the proper fuses to implement the logic design. As simple as this process sounds, many programmers require manual translation of the logic design, akin to manually translating assembly language software into machine language. The programming steps are virtually the same as those taken when programming PROMs. To start the programming sequence, first enter a device code, then programming can begin (see section 3.4).

When a manufacturer ships a programmable logic device, the manufacturing cycle is only partially complete. Because the device is not fully manufactured until it is programmed, the fault coverage of the manufacturer's final test is limited. Therefore, the programmer must be able to perform some type of functional testing. This is accomplished in the PLDS by using structure testing and Logic Fingerprint™ testing. Refer to section 1.4.3 for additional information.

### 1.4.1 DATA DEVELOPMENT

The first step in data development is to design the logic using some convenient language. The PLDS contains two adapters to enable you to develop your logic design: PALASM design adapter and H&L design adapter. The PALASM adapter is to be used with PAL-type devices and translates Boolean equations into fuse data. The H&L design adapter is used with IFL-type devices and translates logic function tables into fuse data.

These design adapters are used in conjunction with a terminal that is connected to the programmer. The PLDS enables you to specify a device and input the design in the form of Boolean equations or logic function tables from the terminal via RS-232 port in the programmer.

The adapter firmware translates your design into a fuse pattern and, optionally, test vectors. This fuse pattern and test data are resident in the programmer's RAM. Thus, the design adapter can be removed from the LogicPak™, and an appropriate P/T adapter can be installed allowing the fuse pattern to be programmed directly into the device and automatically tested. The detailed operating procedures for data development are provided in the PALASM and H&L design adapter manuals.

If a fuse pattern is generated on a host system, it must use fuse numbers specified according to logic diagrams in the P/T adapter manual and transmitted to the programmer in the JEDEC (Joint Electron Device Engineering Council) format (see appendix A). Data I/O uses the JEDEC Logic Device Translation Format (number JC-42, 1-81-62) for serial data input and output with the LogicPak™. The only exception to this is when you are using a Signetics H&L design adapter, in which case data transfer can also occur in the Signetics H&L logic format.

An alternate method of specifying the fuse map is to manually enter the fuse number and state for every fuse in the device. Each P/T adapter manual contains logic diagrams for the devices in its repertoire. These are the same as those in the device manufacturers' data books, but the fuse number has been added. Although tedious, fuse numbers and states can be entered manually into the programmer's data RAM from the programmer's keyboard or from a terminal. This method usually will be used only for editing fuse data because it is a long process with room for error.

With a P/T adapter, fuse data can be entered into the programmer's RAM by loading from a master device. Blank devices can then be programmed using the same P/T adapter, or other manufacturers' functionally equivalent second-source devices can be programmed by installing the appropriate P/T adapter. Remember that a device that has its security fuse programmed cannot be used as a master because its fuses cannot be read.

A different programmer RAM fuse map was used in some previous Data I/O logic device programming modules (950-0800, 919-1427, and 919-1542). If you have paper tapes or files that you prepared to use with these modules, you must prepare new files for use with the LogicPak™ unless you used the Signetics H&L logic format. The preferred translator format is the JEDEC format, but a Signetics translator is available. The 950-0104 and 919-0045 modules use a memory map that is compatible with the LogicPak™. The 919-1427 pak used a fuse map generated to simulate a 512 x 4 PROM. Serial data were entered using a standard PROM data translation format. Again, tapes must be regenerated for use with the LogicPak™ because of the different correspondence between fuses and bits in RAM. The fuse maps have been changed to allow for the programming of functional second-source devices from the same fuse map loaded in RAM and to avoid gaps (previously called phantom fuses) in the fuse maps.

*NOTE*

*If master devices are used as a data source, the above considerations do not apply. A master device can be used as the transfer medium between different fuse maps. Use the old pak (old fuse map) to program a master device, then install the LogicPak™ and load the master. The new LogicPak™ fuse map will then reside in RAM.*

### Sum-Check

After fuse data have been loaded into the PLDS from the serial port or the master device, the programmer calculates the sum-check of the fuse data (figure 1-3) and displays it (when in terminal mode, it will be displayed on the terminal). The sum-check, which is used to verify the integrity of data transfers, is a summation of eight-bit bytes of fuse data expressed as a four-digit hexadecimal number (see figure 1-3 or 3-14).

| HEXADECIMAL DATA | BINARY DATA |
|---|---|
| 84 | 10000100 |
| C1 | 11000001 |
| 62 | 01100010 |
| 24 | 00100100 |
| 01CB | 0000 0001 1100 1011 |
| ↑ Sum-check in hexadecimal notation | ↑ Sixteen-bit binary sum-check |

**Figure 1-3. Sample Sum-Check Calculation**

If data were loaded through the serial port and a sum-check was sent with it, the programmer will compare the sum-check with its own calculation. If they agree, the correct sum-check is displayed. If they do not agree, the programmer will signal an error. Data from the serial port will also be checked for correct parity if the program parity switch is on (see your programmer manual).

Data from a master device are loaded into RAM by installing the correct P/T adapter on the LogicPak™ and performing a load operation. Any information in the source buffer (Boolean equations, function tables, or test vectors) will not be affected.

### Editing Features

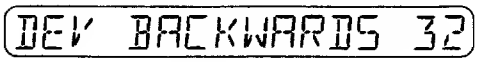
After data have been developed or loaded into programmer RAM, the PLDS provides editing features that allow you to modify the fuse pattern. These are accessed through select functions as described in section 3.5.

### Data Lock

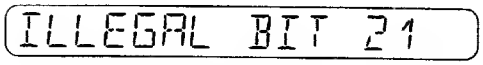
The System 19, 100A, and 29A programmers have a data-lock feature that allows you to limit the accessibility of your RAM data. This is useful in production environments to protect the integrity of your RAM data. Refer to the programmer manual for information on data lock.

## 1.4.2 PROGRAMMING

After data development, the next step in the programming sequence is programming the logic devices. The LogicPak™ applies the manufacturer's specific algorithms to blow fuses in the logic device according to the fuse pattern in the programmer RAM. Once you key in the operation on the programmer, programming is automatic and starts with a series of tests: backward device test, illegal bit test, and blank check. During the backward device test, the programmer automatically checks the device's orientation in the P/T adapter socket and displays an error if it is inserted backwards. The 29A displays:

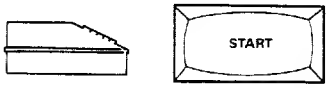


The illegal bit test checks for previously programmed bits in a nonblank device that cannot be programmed according to the fuse pattern in RAM. If illegal bits exist, the 29A displays:

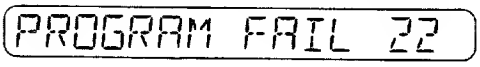



During the blank check, the programmer searches the device for programmed bits. If any are found (and the bits are legal), the programmer will signal the operator and the 29A displays:


Nonblank parts can be over-programmed by again pressing




If the device passes these tests, data are transferred from the programmer RAM to the LogicPak™ one byte (eight fuse states) at a time. The LogicPak™ then applies the programming pulses to the first fuse and tests its condition. If the fuse fails to program, the 29A displays:



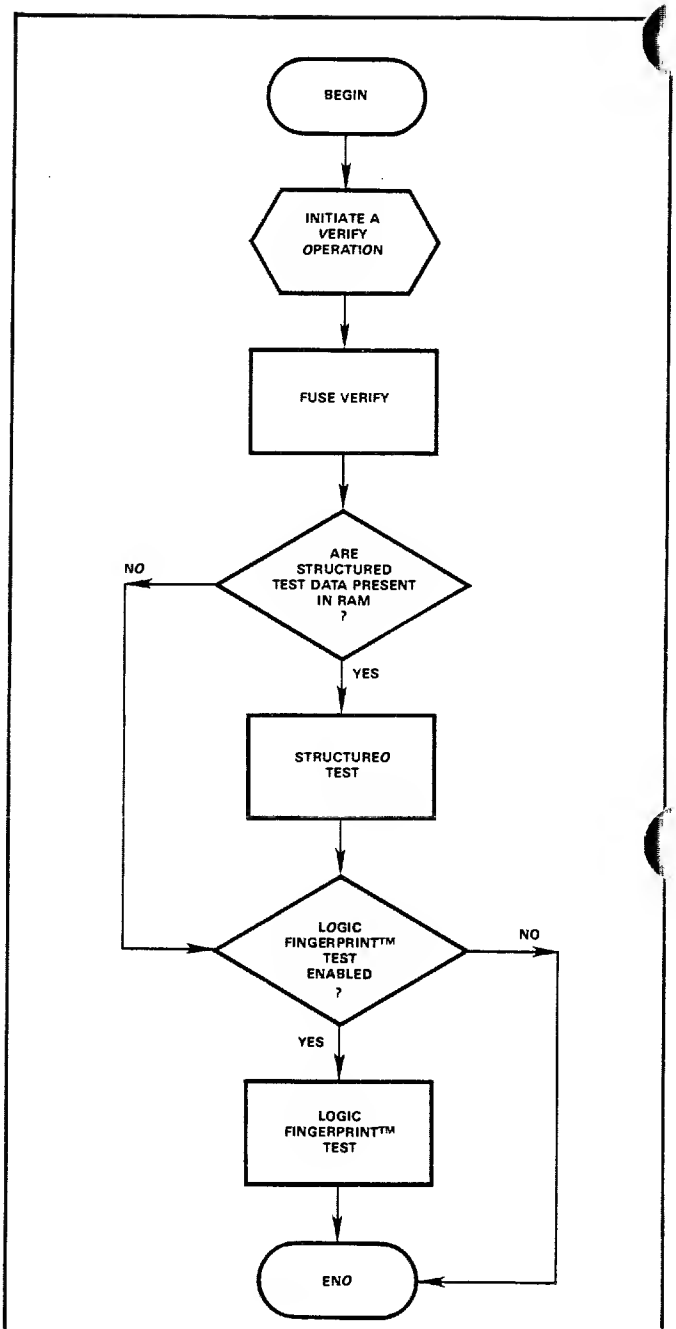

Otherwise, programming proceeds to the next fuse until all have been programmed (With some P/T adapters, programming algorithms vary and may only display a verify error.). Refer to the P/T adapter manuals for manufacturer-specific programming algorithms and waveform pictures.

### 1.4.3 FUNCTIONAL TESTING

The next step in the programming sequence is functional testing, which verifies that the device will perform as intended. Figure 1-4 shows the sequence used to verify and test devices with the LogicPak™; see section 3.4.6 for details on the verify options.

**Fuse Verify.** The fuse verify ensures that the fuse pattern in the device and the programmer RAM are the same, which would indicate that all fuses have been correctly programmed. If the fuse pattern of the device and the programmer RAM correspond, the verify operation will display the sum-check of the RAM data.

During a fuse verify, the LogicPak™ compares the fuse pattern of the programmed device, one fuse at a time, with the fuse pattern in the programmer data RAM. When fuse states in RAM do not correspond to those in the

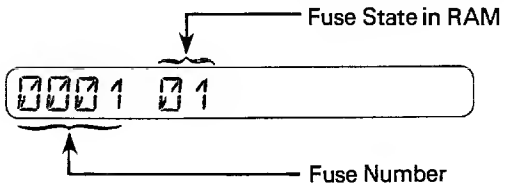



*NOTES*

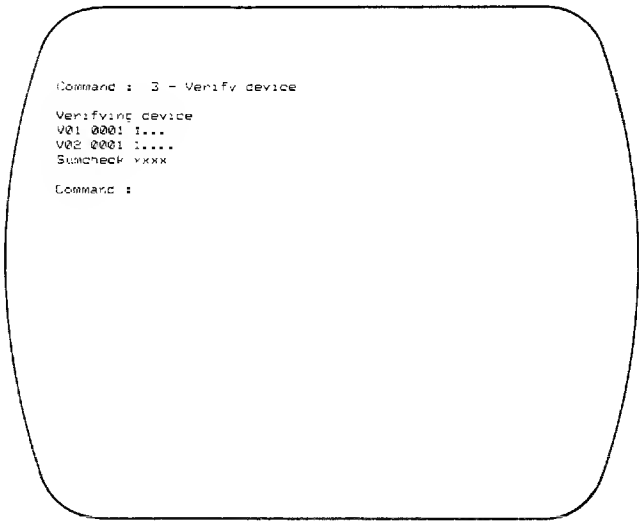
1. *This sequence will occur automatically after a program operation. To verify a previously programmed device, the VERIFY key on the programmer keyboard or a remote control verify command is used.*
2. *The structured test occurs (if data are present) before Logic Fingerprint™ test. This is useful to initialize sequential devices so the Logic Fingerprint™ test is more effective.*

**Figure 1-4. Verifying and Testing Procedure Flowchart**

programmed device, the PLDS will display the fuse number, the fuse state in RAM, and whether the first or second pass verify produced the failure. For example, the 29A will display:




The terminal will display:




XXXX is the sum-check.

Most of the logic paths of a programmable logic device are not tested by the fuse verify. Therefore, a fuse verify will not guarantee that the device will perform its intended function; however, it is a necessary step to ascertain whether or not the device has been programmed correctly.

**Structured Test.** The structured test is optional. A structured test will be performed on the device automatically after programming and during a verify operation only if there are structured test data present in the programmer data RAM. If no data are present, only a fuse verify and a Logic Fingerprint™ test (if enabled) will be performed.

The structured test supplements the Logic Fingerprint™ test. It lets you enter test vectors that stimulate and read device pins, guaranteeing that specific states will be tested. It can also be used to initialize devices for the Logic Fingerprint™ test. The structured test enables you to uniquely define the inputs and test for desired outputs. The LogicPak™ applies those inputs and verifies that the desired outputs appear. The structured test cannot be performed unless there are structured test vectors in RAM. With the vectors present, the structured test can be disabled by selecting verify option "1" (see section 3.4.6).

When a device fails a structured test while in the terminal mode, the terminal will display the vector test number and output pin number that failed to show the specified levels. When using the programmer, the 29A will display an error code (see section 4, table 4-1). The structured test requires that you write your own test vectors and input them to the programmer RAM, or that you use data development aids to generate the test vectors from specified inputs, outputs, or function tables. Although a structured test guarantees that certain specified states have been tested, writing vectors is a time-consuming task that requires a thorough understanding of the designed function of the device under test.

Test vectors can be developed efficiently with the PALASM design adapter. This design aid allows you to enter equations and function tables. Firmware compares the equations and function tables during an operation called "Simulate Function Table." If the equation and function tables are valid, the firmware generates structured test vectors. Data I/O recommends this method as it is the fastest and easiest way to produce structured test vectors. If vectors are developed in any other way, they must be manually input to the programmer RAM from the terminal.

The structured test requires you to write your test vectors and input them to the programmer data RAM. The programmer RAM is limited in the number of tests it can store (see table 1-3). Each vector tests one specific state; for example, if RAM is limited to 50 states and you require testing 100 unique states, you would test 50 states at one time and use multiple loads and test sequences.

Table 1-3. RAM Capacity for Structured Test Vectors

| PIN ON DEVICE | PROGRAMMER RAM SIZE | | | |
|---|---|---|---|---|
| | 4k | 8k | 16k | 64k |
| 20 | 50 | 150 | 250 | 250 |
| 24 | 42 | 128 | 250 | 250 |
| 28 | 36 | 109 | 250 | 250 |

The structured test can be used to: 1) apply all possible input states to a combinational device, 2) force a sequential device through all its state transitions, and 3), because the structured test is performed before the Logic Fingerprint™ test, present a series of inputs to a device that will drive it to a known initial state so the Logic Fingerprint™ test can begin from that known state. This is required for registered or sequential devices that may power up in an illegal state. Structured testing is especially useful with sequential devices because pseudorandom testing algorithms do not achieve the same level of test coverage as they do for combinational devices.

Some manufacturers' sequential devices include a preload feature that enables the device to be preset to a known initial state. This preload feature is enabled by specifying special variables in the structured test vector. Refer to section 3.5.7 for specific structured test vectors. Sequential devices without this feature will need to be designed and programmed so that structured vectors can force them into a known initial state. This means that you must design the registered (sequential) device so that it can be tested, then write structured test vectors that initialize the device for testing.

Once written, structured vectors can be stored as part of the JEDEC formatted file (see appendix A).

**Logic Fingerprint™ Test.** Of all testing methods available in logic device programmers, the Data I/O Logic Fingerprint™ test is the easiest to use and is flexible enough to be applicable to most logic devices. The Logic Fingerprint™ test is similar to signature analysis. To test a device using the Logic Fingerprint™ test, it is necessary for the LogicPak™ to first learn a signature (test-sum) from a known-good device. Subsequently programmed devices will be tested, and their test-sums will be compared against the reference test-sum. If the test-sums match, the device has passed the Logic Fingerprint™ test.

Figure 1-5 shows a block diagram of the Logic Fingerprint™ test. The test uses a shift register as a pseudorandom pattern generator to stimulate the device under test with test vectors. Depending on how the device is programmed, it responds to these inputs with some set of outputs. These outputs are used in conjunction with the inputs to generate the next vector.

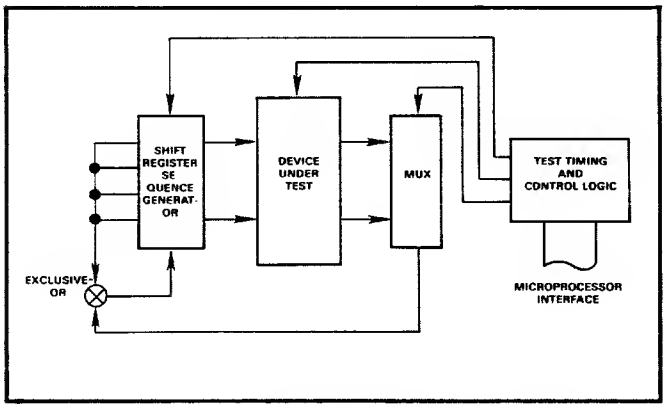



**Figure 1-5. Logic Fingerprint™ Test Block Diagram**

The outputs are fed back throughout the pseudo-random sequence generator, 128,000 times for each cycle of the Logic Fingerprint™ test. The contents of the register after the test is the test-sum unique to the programmed device. The test-sum is then automatically compared to the stored reference test-sum.

The LogicPak™ learns the reference test-sum from a known-good master device during a load operation. If already known, it can be loaded via the serial port along with programming data (in the JEDEC format) or entered from the programmer keyboard or terminal. If a structured test is being used to initialize devices for the Logic Fingerprint™ test, the structured test will be performed upon a load operation prior to learning the test-sum. If the device fails the structured test in load, an error code will be displayed (see section 4, table 4-1).

The Logic Fingerprint™ test is enabled by entering the number of test cycles desired. The default number of cycles is zero, which disables the test. Any number of cycles up to 99 can be entered; however, each additional cycle will increase the time of the Logic Fingerprint™ test. Each cycle takes 3 to 6 seconds, depending on the device type. Only 1 to 8 cycles are necessary in most cases. More than one cycle will be used most often with sequential devices that require more than 128,000 testing cycles for a satisfactory test.

The operator can specify the starting vector for the Logic Fingerprint™ test. The starting vector could be used in two instances: 1) to initialize a sequential device that may not power-on reliably to a known initial state or 2) to test a device that will not respond to the default starting vector of all bits set to zero. Refer to the P/T adapter manuals for examples. The starting vector will enable the Logic Fingerprint™ test to begin from a known initial state. The starting vector can be up to 28 bits long, depending on the number of pins on the device. A 20-pin device will have a 20-bit starting vector, and a 24-pin device, 24-bit vector.

The pseudorandom nature of the input vectors can cause some devices in some programming circumstances to fail the Logic Fingerprint™ test by giving nonrepetitive test-sums. THIS DOES NOT NECESSARILY INDICATE A FAULTY DEVICE, but may be an indication that the device is subject to the Logic Fingerprint™ test limitations. Logic Fingerprint™ test limitations are described in each P/T adapter manual for each manufacturer's devices. IT IS VERY IMPORTANT THAT YOU READ AND UNDERSTAND THESE LIMITATIONS.

The Logic Fingerprint™ test can be disabled by setting the number of Logic Fingerprint™ test cycles at zero (This is the normal default). If the Logic Fingerprint™ test is not suitable for a specific device, it can always be tested by using the structured test.

**Self-Testing.** The LogicPak™ performs a series of self-tests on its hardware to assure the functionality of the module. A failure will display an error code (see section 4, table 4-1). To isolate the problem to its source, refer to section 4.4 on troubleshooting.

Because of the internal analog feedback capability in the LogicPak™, voltage levels can be measured so that a "go/no-go" test can be performed. Programmable voltage levels are tested when device-related operations are executed. The TTL (transistor-transistor logic) levels are tested when the family and pinout codes are entered with the system 29A and 100A programmers and when the device-related operations are executed with the system 19 programmer. Testing is accomplished under software control by first setting test levels and then reading back

these levels through analog comparators. The comparator output levels are then verified against predetermined values.

When the programmer is powered up, the Logic Fingerprint™ self-test is done by testing an open socket. The LogicPak™ compares the open-socket test-sum against a predetermined test-sum stored in the firmware. This guarantees that all Logic Fingerprint™ test circuitry is functional. These tests are performed during calibration operations; see section 4 for the diagnostic steps.

The self-test routines will pinpoint any failures immediately and flag with an error if there is a problem. Therefore, you can be confident that your equipment is in proper operating condition. All the testing features Data I/O has incorporated into the PLDS give you the security of knowing you will have the highest possible programming yields. The hardware self-tests assure you your equipment is capable of programming and testing. The device tests (backwards device, illegal bit, blank check, fuse verify, Logic Fingerprint™, and structured tests) assure you that your devices are programmed correctly and test functionally. Together these tests result in the maximum programming yield possible with any commercial programmer.

## 1.5 LOGICPAK™ OVERVIEW

The Data I/O 303A LogicPak™ is the base unit of the PLDS. It contains all the common electronics and firmware for programming and testing logic devices. Any electronics or firmware unique to a specific device family, or device within a family, are resident in P/T adapters that plug into the LogicPak™. Families with more than one pin number series (e.g., PAL 20 and PAL 24) have sockets to accommodate each pin count.

Software tables store values for programming variables including pinouts, voltage levels, and timing. When you choose the family and pinout codes for a particular device, the programmer uses information in these tables to assemble a specialized programming routine in scratch RAM. This method allows high-speed operation with minimum firmware overhead.

To increase flexibility in waveform generation, digital-to-analog converters (DAC) control all major power supplies, with several rise and fall times selected by software.

In addition to its programming and testing capabilities, the LogicPak™ is the base unit for design adapters. These adapters permit high-level Boolean equation or function-table entry.

## 1.6 APPLICATIONS

A complete list of all currently available logic devices and the hardware necessary for data development, programming, and testing of each is included with each adapter manual. As Data I/O increases the capabilities of the LogicPak™ to program new or additional devices, firmware updates will be available for existing adapters to add new devices to existing device families. New adapters may also be added to the PLDS to accommodate new device families.

## 1.7 SPECIFICATIONS

The LogicPak™ receives its power from the programmer power supplies. Programming waveforms are generated from programmer supplies using DACs controlled by the programmer's microprocessor. The controlling firmware is located both on a circuit board in the LogicPak™ and in the P/T adapters or design adapters. The physical and environmental specifications of the LogicPak™ are:

- altitude (operating): sea level to 3 km (10,000 ft)
- humidity (operating): 90% maximum (noncondensing)
- humidity (storage): 95% maximum (noncondensing)
- temperature (operating): 5 to 45°C (41 to 113°F)
- temperature (storage): −40 to 70°C (−40 to 158°F)
- weight: 1.6 kg (3 lb, 8 oz)
- dimensions: 17.9 x 17.3 x 21.7 cm (7.05 x 6.81 x 8.54 in.)

## 1.8 FIELD APPLICATIONS SUPPORT

Data I/O has field applications engineers throughout the world. They can provide additional information about interfacing Data I/O products with other systems and answer questions about your equipment.

These engineers are located within the United States at the addresses listed in the back of this manual. For international applications support, contact your nearest Data I/O representative.

## 1.9 WARRANTY

The Data I/O LogicPak™ is warranted against defects in materials and workmanship. The warranty period of 90 days begins when you receive the equipment; the warranty card inside the back cover of this manual explains the length and conditions of the warranty. For warranty service, contact your nearest Data I/O service center.

## 1.10 SERVICE

Data I/O maintains service centers throughout the world, each staffed with factory-trained technicians to provide prompt, quality service. A list of all service centers is located in the back of this manual.

## 1.11 ORDERING

To place an order for equipment, contact your Data I/O sales representative. Orders for shipment must include:

- a description of the equipment (see the latest Data I/O price list or contact your sales representative for equipment and part numbers)
- purchase order number
- desired method of shipment
- quantity of each item ordered
- shipping and billing address of the firm, including ZIP code
- name of person ordering the equipment.

# SECTION 2

# INSTALLATION

## 2.1 INSPECTION

The 303A LogicPak™ was tested both electrically and mechanically before it was shipped and was carefully packaged to prevent shipping damage. It should, therefore, arrive free of any defect, without marks or scratches, and in perfect operating condition. Carefully inspect the instrument for any damage that may have occurred in transit; if you note any damage, file a claim with the carrier and notify Data I/O. Also, check that the following equipment is present:

- 303A LogicPak™ (950-1942)
- Instruction Manual (10-950-1942).

## 2.2 LOGICPAK™ INSTALLATION

The LogicPak™ may be installed and removed with the programmer's power on; this feature allows you to retain data in RAM during module changes. If the programmer power is turned on before the LogicPak™ is installed, you will hear a beep until the LogicPak™ with an adapter is installed.

*NOTE*

*Voltage transients can cause device damage. If a P/T (programming/testing) adapter is installed in the LogicPak™, be sure that all sockets are empty when:*

- *switching power on or off*
- *installing or removing the LogicPak™ or adapter.*

To install the LogicPak™:

1. Slide the LogicPak™ into the opening in the programmer (see figure 2-1a).
2. Tilt the LogicPak™ up and gently push it back to hook the flange of the LogicPak™ over the back edge of the programmer opening (see figure 2-1a).
3. Lower the LogicPak™ into position (see figure 2-1b).
4. Press gently on the front edge of the LogicPak™ to ensure a good connection (see figure 2-1c).

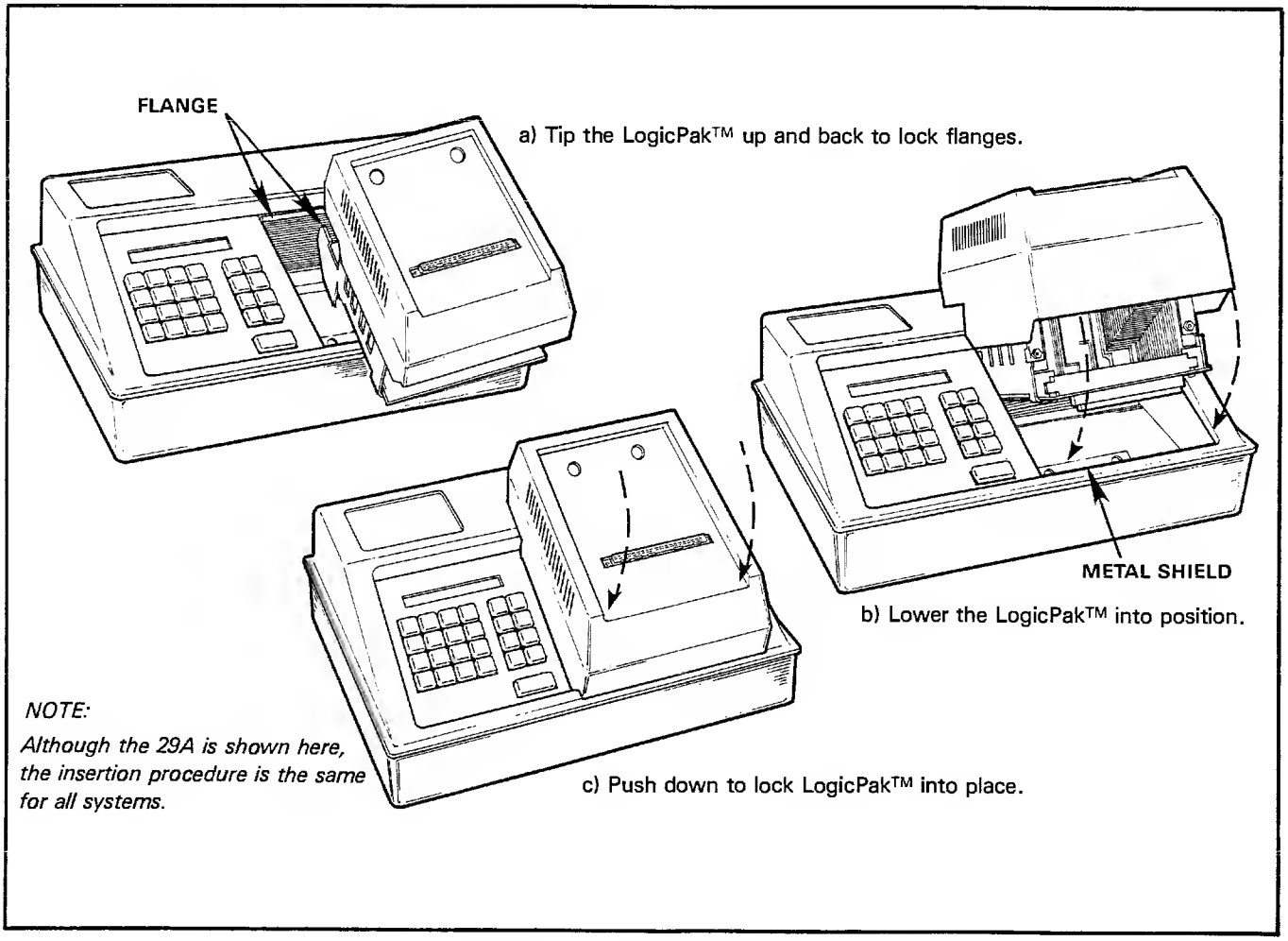



**Figure 2-1. LogicPak™ Installation**

## 2.3 LOGICPAK™ REMOVAL

To remove the LogicPak™:

1. Check to make sure the programmer is not in the process of an operation. If it is, wait until the operation is complete: the action symbol on the 29A programmer display will disappear (see your programmer manual). In the terminal mode, press ESC (escape).
2. Check to make sure a device is not in a socket. If one is in a socket, remove it as described in section 3.4.3.
3. Grasp the sides of the LogicPak™ and lift.
4. Pivot the LogicPak™ toward the rear of the programmer to unlock the flanges, then lift out.

## 2.4 ADAPTER INSTALLATION

To insert all adapters for the LogicPak™:

1. Check to make sure a device is not in a socket. If a device is in a socket, remove it as described in section 3.4.3.
2. Align the guide pins on the underside of the adapter with the guide pin holes on the LogicPak™ (see figure 2-2).
3. Gently set the adapter on the LogicPak™. Make sure the guide pins line up with the guide pin holes.
4. Firmly press down on the front edge of the adapter to lock the connector pins into the connector receptacle (see figure 2-2).

## 2.5 ADAPTER REMOVAL

**CAUTION**

**BEFORE REMOVING THE ADAPTER, press ESC from the terminal, or from the programmer front panel, press the KEYBOARD key (on the System 19) or the VERIFY key (on the 100A or 29A). Because the processor in the programmer executes firmware resident in the adapter, these precautions must be taken before removing the adapter from the LogicPak™ to prevent a program interrupt or loss of RAM data.**

To remove the adapter:

1. Ensure that the programmer has completed the current operation.
2. Ensure that a device is not in a socket.
3. While holding down the LogicPak™, grasp the handle and gently remove the adapter.

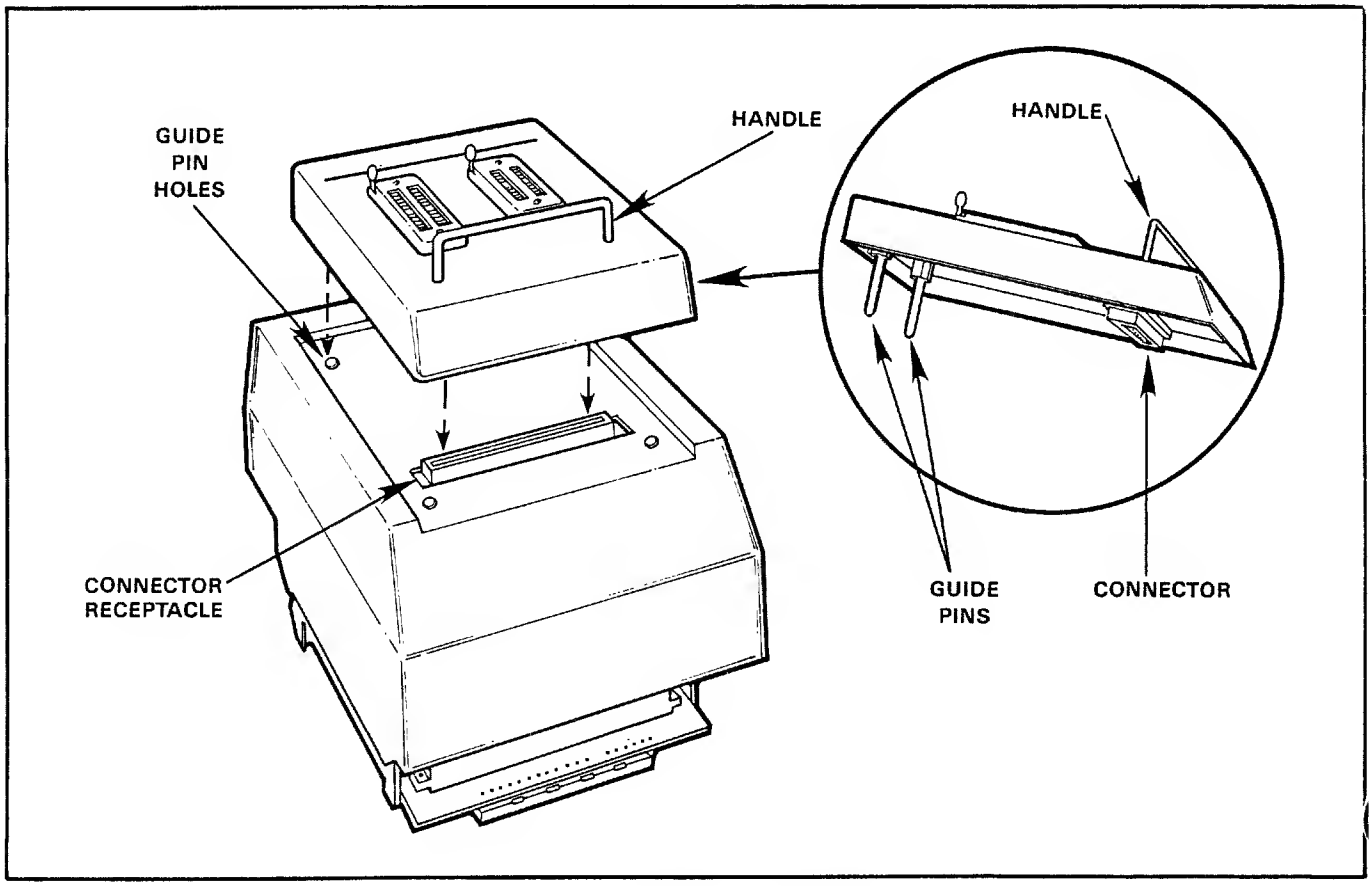



**Figure 2-2. Adapter Installation**

## 2.6 SERIAL INTERFACING

The LogicPak™ uses the RS-232C serial interface of the programmer for interaction with a terminal and for communication with a host computer. Figure 2-3 illustrates some typical interconnection methods.

The five-wire handshake interconnection may be used to upload (data flow from LogicPak™ to host) or download (data flow from host to LogicPak™) data at any supported baud rate with a host system that recognizes the five-wire protocol (see figure 2-3a).

Software handshake (CTRL S, DC3, or ASCII hex 13 to stop transmission and CTRL Q, DC1, or ASCII hex 11 to resume) may be used with either the five-wire or the three-wire interconnection (figure 2-3a or b), but will be recognized by the LogicPak™ only while uploading. CTRL Y (EM, or ASCII hex 19) may be sent by the host to terminate an upload.

*NOTE*

*Due to processing overhead, I/O overrun errors may occur when downloading JEDEC files (see section 3.5.9) over a three-wire link at baud rates above 4800.*

The control characters described above have a similar effect when entered from a terminal: CTRL S AND CTRL Q will halt and resume output to the display, and CTRL Y will terminate a command during display output.

*NOTE*

*An ESC character (ASCII hex 1B) received at any time will terminate all LogicPak™ operations immediately and return control to the programmer.*

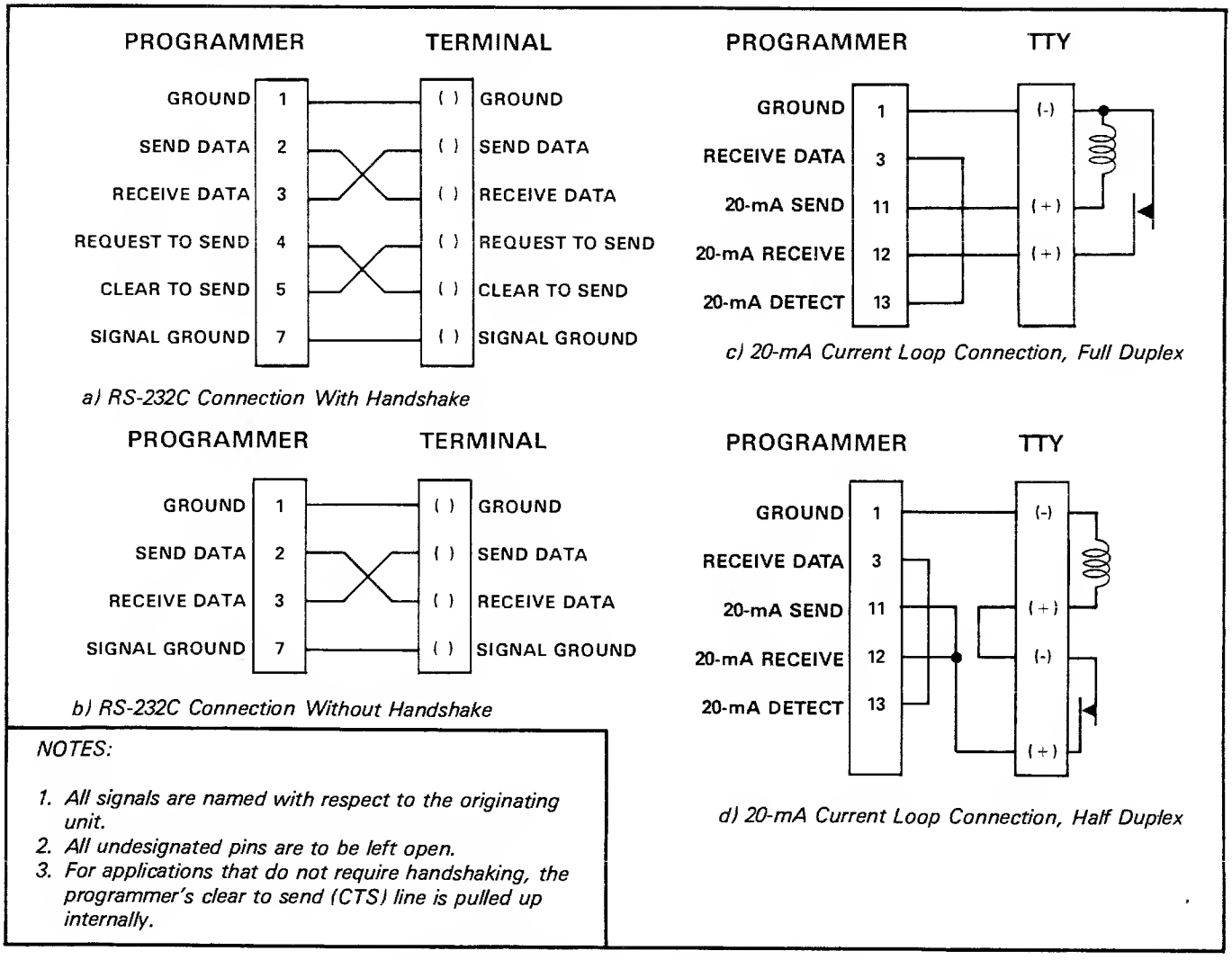



**Figure 2-3. Sample Interconnection Methods**

The LogicPak™ will transmit the line feed character (LF, or ASCII hex 0A) and a variable number of NULL characters (ASCII hex 00) following every RETURN character (ASCII hex OD), based on the null count entered using select function D9 on the System 19 or 29A (see table 2-1).

**Table 2-1. Null Count/Send Chart**

| Null Count[a] | LF Sent? | Number of Nulls Sent |
|---|---|---|
| 00-7F | Yes | Same as null count |
| 80-FE | No | Null count minus 80 hex |
| FF | No | None |

[a] See your programmer manual for details on entering the null count.

*NOTE*

*The NULL count option is not available when the LogicPak™ is installed in a Model 100A. A line feed and four NULLs are always sent.*

To set the baud rate, parity, and stop bits, refer to your programmer manual; we recommend a transmission rate of 9600 baud.

*NOTE*

*Be sure the programmer power is off before setting the parity and stop bits.*

To set up your terminal:

1. Disable any special terminal functions that use CTRL Z, CTRL C, CTRL Y or CTRL P.
2. Select the cursor mode that deletes the character in the current cursor position (i.e. destructive cursor). This will vary from terminal to terminal.
3. The default line mode of the LogicPak™ varies with its system as follows:
   - Full duplex: System 19.
   - Half duplex: System 19, 29A and 100A.

   The line mode of the LogicPak™ may be changed at any time (see section 3.5.12).

## 2.7 REPACKING FOR SHIPMENT

If the LogicPak™ is to be shipped to Data I/O for service or repair, attach a tag to it describing the work required and identifying the owner. In correspondence, identify the unit by part number, revision level, and the name of the unit. If the original shipping container is to be used, place the LogicPak™ in the container with the appropriate packing materials, and seal the container with strong tape. If another container is used, be sure that it is a heavy carton, wrapped with heavy paper or plastic; use appropriate packing material, and seal well with strong tape. Mark the container "DELICATE INSTRUMENT" or "FRAGILE."

# SECTION 3

# OPERATION

## 3.1 OVERVIEW

To program a logic device, you must first load the desired state of the device fuses into the programmer RAM. With Data I/O's Programmable Logic Development System (PLDS) you may input this fuse information in several forms: H&L programming tables for IFL devices, Boolean equations for PAL devices (PALASM), and decimal fuse number and state. The PLDS translates any one of these inputs into the LogicPak™ fuse map such as the one shown in figure 3-1; see section 1.4.1 for a detailed description of the LogicPak™ fuse map. The programmer then blows the fuses in the logic device according to the fuse map. The procedure used to blow the fuse is called the programming algorithm and is stored in the P/T adapter firmware. The device is verified and functionally tested after programming.

The LogicPak™ can be used in 29A, System 19, or 100A programmers of any configuration (see section 1.3 for hardware and firmware compatibility levels required). The LogicPak™ can obtain data from three sources: a master device, a serial port, or from the keyboard (see figure 3-2).

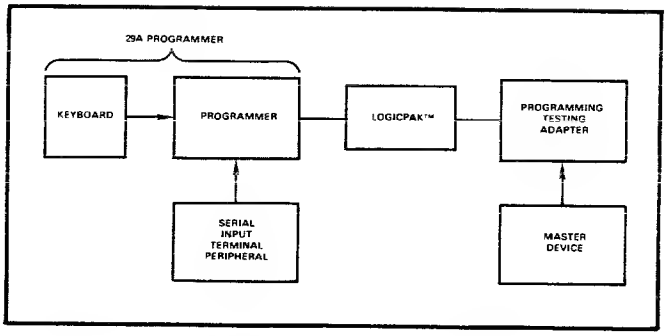



**Figure 3-2. Loading Data Via the Serial Port or Master Device**

The procedures used to perform basic operations with your LogicPak™ are described in this section. Wherever possible, key sequences have been included to use your LogicPak™ with a 29A Universal Programmer updated to Rev C firmware (read section 1.3 carefully to determine your programmer's firmware revision level). Refer to your programmer manual for key sequences for the System 19 and 100A programmers.

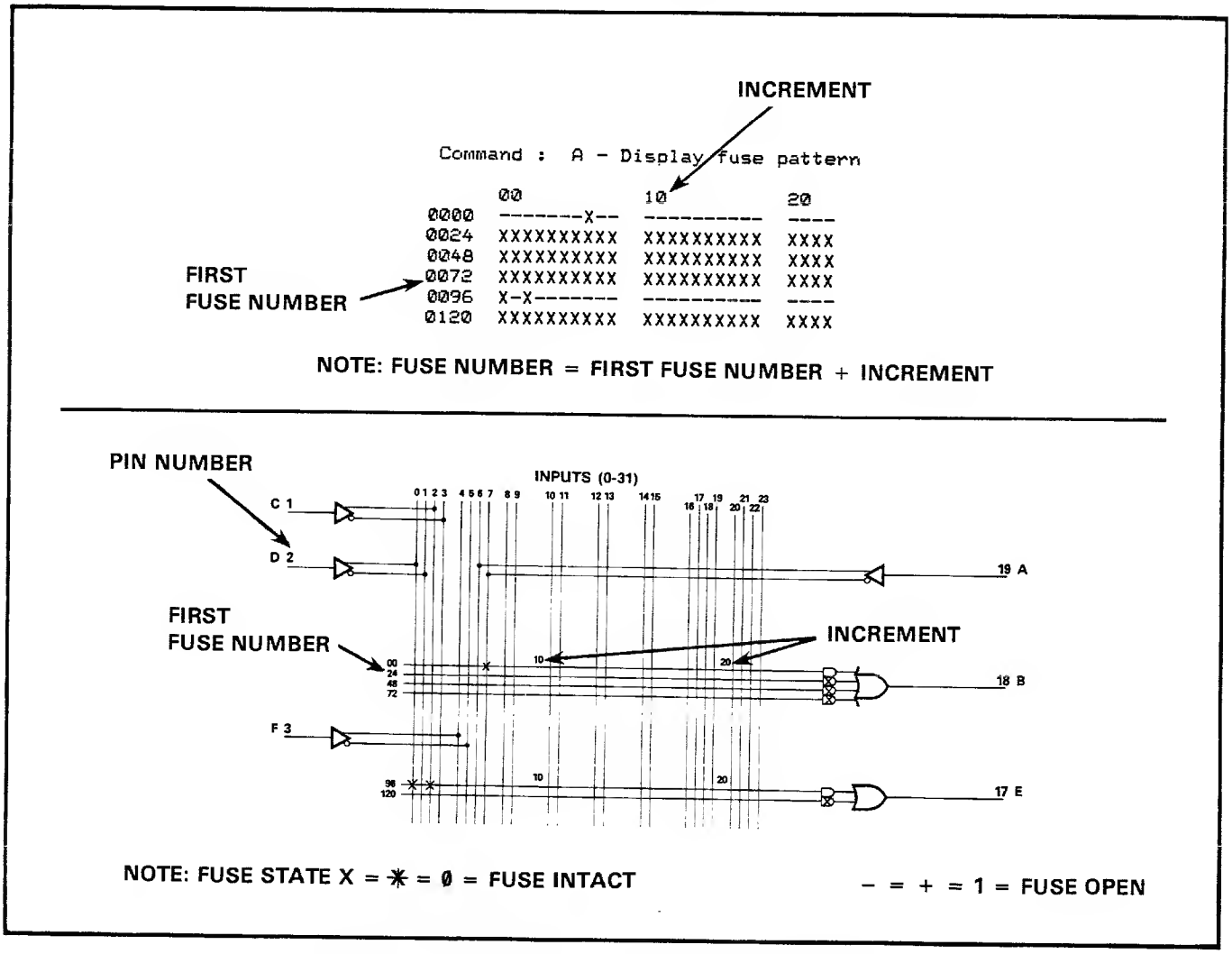



**Figure 3-1. Fuse Map and Corresponding Device Diagram**

## 3.2 POWER UP

*NOTE*

*If the LogicPak™ with an adapter installed is not in the programmer before power is turned on, you will hear a beep until the LogicPak™ is installed.*

*When power is applied, the programmer will perform an automatic self-test routine (see section 1.4.3). When the self-test routine is complete, the programmer will signal its readiness (see your programmer manual).*

To turn the programmer on:

1. Check to make sure a device is not in a socket. If a device is in a socket, lift up the lever (on the upper left of the socket; see section 3.4.2), then gently lift the device out of the socket.
2. Plug the AC power cord into the power outlet.
3. Lift the power switch up to the ON position (see figure 3-3).

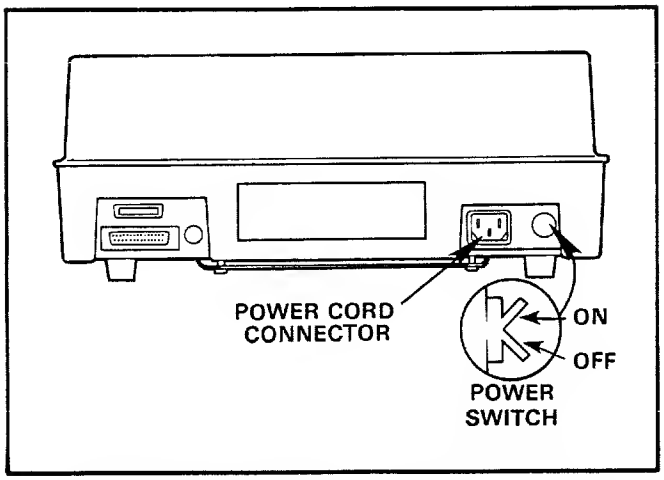



**Figure 3-3. Programmer Power Switch Location**

## 3.3 POWER DOWN

**CAUTION**

**Do not turn the power off when a device is in a socket; voltage transients may damage the device.**

To turn the programmer power off:

1. Check to make sure the programmer is not in an operation process. If it is, wait until the operation is complete.
2. Check to make sure a device is not in a socket. If a device is in a socket, remove it as described in section 3.4.3.
3. Push the power switch down to the OFF position (figure 3-3).

## 3.4 BASIC DATA TRANSFER OPERATIONS

The basic operations that can be accomplished with the LogicPak™ and 29A Universal Programmer are:

- develop data (described in section 1.4.1 and design adapter manuals),
- load RAM with master device data (described in section 3.4.4),
- program a device with RAM data (described in section 3.4.5),
- verify RAM data against the device data (described in section 3.4.6),
- functionally test device (described in section 1.4.3).

The following sections describe device-related operations with the PLDS using a P/T adapter. Most setup procedures specify that you enter the family code and pinout codes because Data I/O recommends that you develop the habit of entering these codes when prompted by the equipment. However, if you are using a design adapter, you will be able to perform non-device-related operations without entering the family code and pinout codes; for example, with the PALASM design adapter, Boolean equations can be entered into the PALASM editor without family code and pinout codes. (Family code and pinout codes are automatically generated upon execution of "translate source equations" in the PALASM mode.)

If the programmer has been used to program PROMs, or contains data in RAM for some other reason, the fuse pattern developed for logic devices could be adversely affected or option parameters could be inadvertently set. Therefore, execute the "clear RAM" select function (see programmer manual), or switch off the programmer (section 3.3) to clear RAM before beginning operations with the PLDS.

All data transfer or verification operations occur between the programmer's internal RAM and the device or between the RAM and serial port in your programmer. Because the operation procedure to transfer data via a serial port varies among programmers, this manual describes only data transfer using the 29A. For other programmers, refer to the specific operation manual.

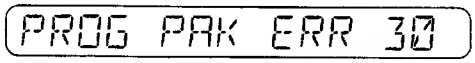
*NOTE*

*An adapter must be installed in the LogicPak™ before any of these operations can be performed (see section 2.4).*

*During copy and verify operations, ADDR and SIZE appear in the 29A prompts. These correspond to starting address and block size, respectively. These block limits must remain in the default state for logic device programming. An error code (see section 4, table 4-1) will be displayed if these limits are altered. For more detail on these parameters, see your programmer manual.*

### 3.4.1 FAMILY CODE AND PINOUT CODE SELECTION

Any device that can be programmed with the LogicPak™ is specified by a unique combination of a two-digit family code and a two-digit pinout code; these codes are provided in each adapter manual and in appendix B of this manual. Once the codes are entered for a particular device, the LogicPak™ remains set up for any operation with that device until you enter new codes. If invalid family and pinout codes are entered, a beep will sound. In remote control operation,




will be displayed, and the operation will be stopped when you attempt a device operation.

To select the family code and pinout code:

1. Locate the manufacturer and part number stamped on the device.
2. Go to the family code and pinout code table in the appropriate adapter manual and find the manufacturer's name.
3. Go to the column entitled "Device Part Number" and find the number corresponding to the number on the device.
4. Go to the columns labeled "Family Code" and "Pinout Code" to find the code numbers corresponding to the device number for the manufacturer of the device.
5. Enter the family code and pinout code you selected from this table when prompted by the programmer or terminal. An LED (light emitting diode) will light above one of the sockets on the adapter.

### 3.4.2 DEVICE INSERTION

Once you have entered the appropriate family and pinout codes, the LogicPak™ with a P/T adapter installed is ready to accept a device in the socket below the lighted LED.

To install a device:

1. Check to make sure the programmer is not doing an operation. If it is, wait until the operation is complete.
2. Lift the lever on the upper-left side of the socket below the lighted LED (see figure 3-4); the lever will stay in the upright position.
3. Gently set the device in the socket below the lighted LED. Make sure pin 1 of the device is aligned with pin 1 of the socket (upper-left corner); see figure 3-4.
4. A good electrical connection between the device and the socket is essential. To ensure a good connection, push the lever down to lock the device in the socket.

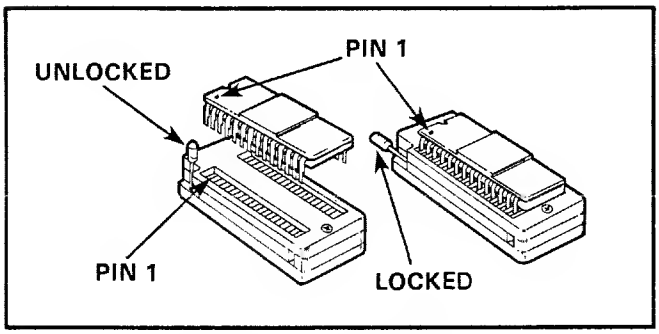



Figure 3-4. Device Installation

### 3.4.3 DEVICE REMOVAL

To remove a device:

1. Check to make sure the programmer is not doing an operation. If it is, wait until the operation is complete.
2. Lift the lever on the left side of the socket; see figure 3-4. The lever will remain in the upright position.
3. Lift the device out of the socket; the LED will remain illuminated.

### 3.4.4 LOAD RAM WITH MASTER DEVICE DATA

#### Front Panel Control

To load the 29A RAM with data from a master device with control from programmer front panel, follow the steps given below.

*NOTE*

*If options are desired (see section 3.5), select options and parameters as needed before proceeding.*

1. 

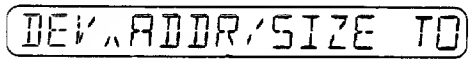
to select the mode.

29A Displays


2. 

to select the source of the data.

29A Displays

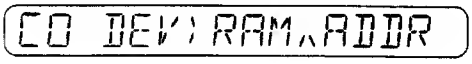


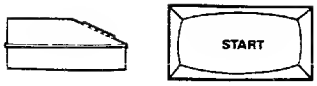

ADDR/SIZE pertains to block limit parameters. These are PROM-related and are not to be used with logic devices. Leave defaults in effect.

3. 

to select the destination for the data.

29A Displays




4. 

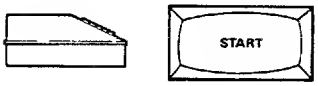

29A Displays


5. Enter the family code and pinout code (see section 3.4.1).

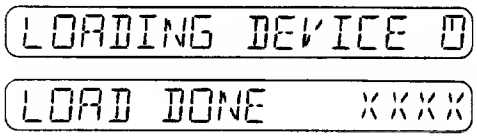
*NOTE*

*The appropriate socket LED will light.*

6. Insert the master device into the appropriate P/T adapter socket. (See section 3.4.2.)

7. 


29A Displays




*NOTE*

*XXXX is the sum-check of the device fuses.*

8. Remove the master device from the adapter socket. (See section 3.4.3.)

### Terminal Control

To load the 29A with data from a master device using the terminal control mode, follow the steps given below.

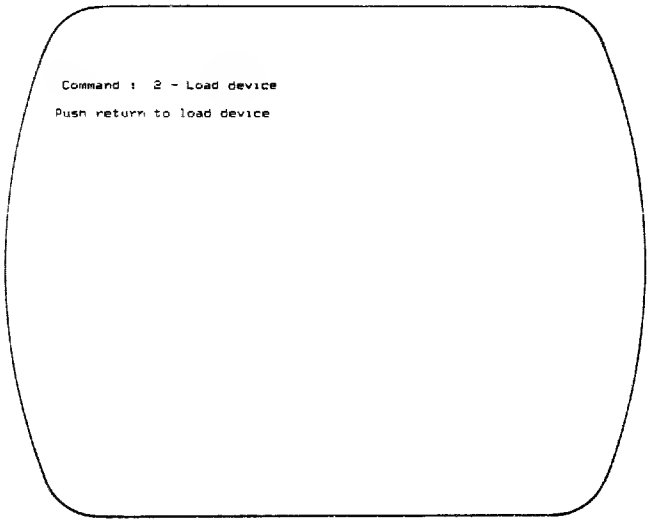
1. Place the system in terminal mode; see section 3.5.1
2. Enter the family codes and pinout codes, if prompted by the terminal.

*NOTE*

*If options are desired (see section 3.5), select options and parameters as needed before proceeding.*

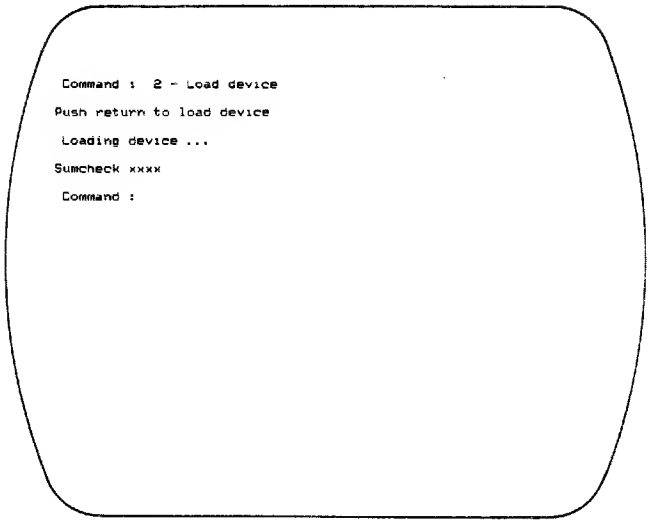
3. 


Terminal Displays




4. 


Terminal Displays




An action symbol will be displayed while the device is being loaded. When loading is complete, the terminal will display sum-check XXXX.

### 3.4.5 PROGRAM DEVICE WITH RAM DATA

*NOTE*

*If options are desired (see section 3.5), select options and parameters as needed before proceeding.*

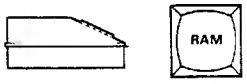
When programming a device, the system performs illegal bit tests and blank checks at nominal $V_{CC}$.

**Front Panel Control**

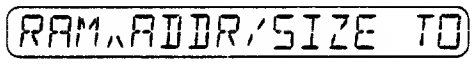
To program a blank device with the data in the 29A RAM with control from the programmer front panel, follow the steps given below.

1. 


29A Displays

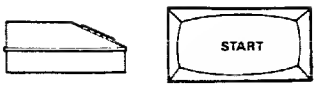

2. 


29A Displays

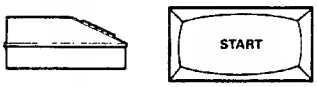



3. 

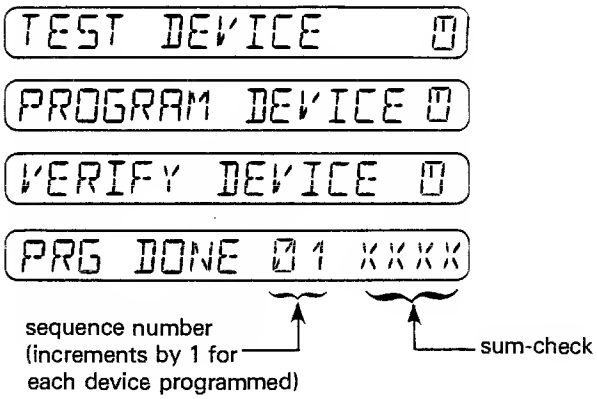

29A Displays


4. 


29A Displays


5. Enter the family code and pinout code if required (see section 3.4.1).
6. Insert the blank device into the adapter socket (section 3.4.2).
7. 


29A Displays




8. Remove the device from the adapter socket (see section 3.4.3).

### Terminal Remote Control

To program a device with 29A RAM data from the terminal control mode:

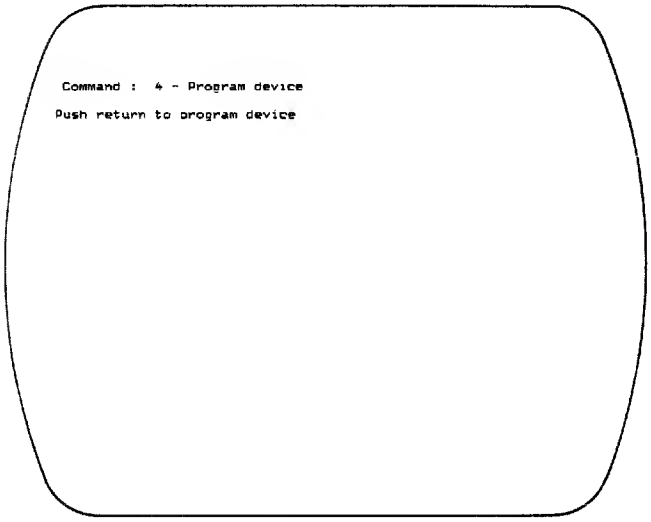
1. Place the system in the terminal mode (see section 3.5.1).
2. Enter the family code and pinout code, if prompted by the terminal.

*NOTE*

*If options are desired (see section 3.5), select options and parameters as needed before proceeding.*

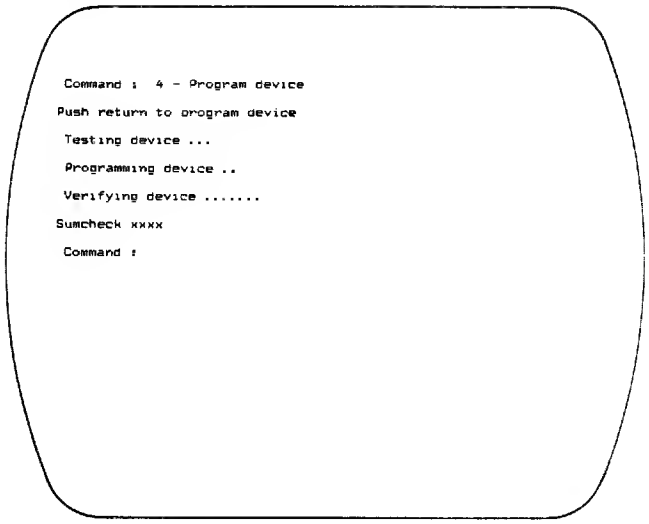
3. 


Terminal Displays




4. 


Terminal Displays




An action symbol will be displayed showing the pretesting, programming and verifying of the part. If no errors occur, the terminal displays sum-check XXXX.

*NOTE*

*XXXX is the sum-check of the device fuses. See figure 3-13 for fuse sum-check calculation.*

### 3.4.6 VERIFY AND FUNCTIONALLY TEST DEVICE

**Front Panel Control**

The verify routine compares the device data to RAM data and performs functional testing, if this option is selected (see section 3.5.5).

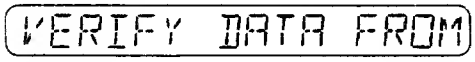
To verify and functionally test a device from 29A front panel control, perform the following steps.

*NOTE*

*If options are desired (see section 3.5), select options and parameters as needed before proceeding.*

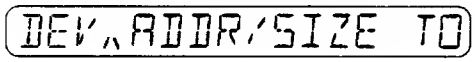
1. 


29A Displays



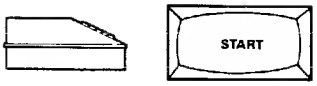

2. 


29A Displays



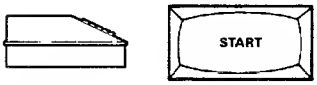

3. 


29A Displays

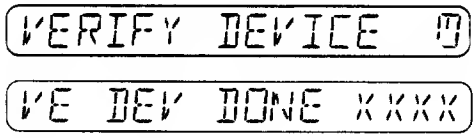

4. 


29A Displays


5. Enter the family code and pinout code if required (see section 3.4.1).
6. Insert the device to be verified and/or tested into the adapter socket (see section 3.4.2).

7. 


29A Displays




*NOTE*

*XXXX is the sum-check of the device fuses.*

8. Remove the master device from the adapter socket (see section 3.4.3).

### Terminal Control

To verify and test a device from terminal control, follow the steps given below.

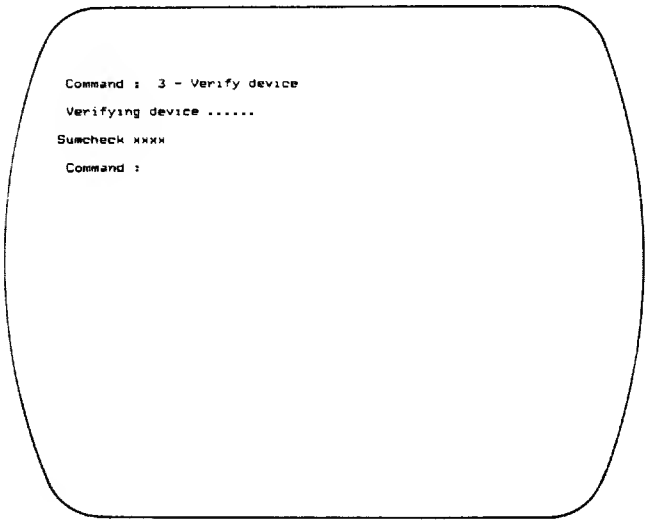
1. Place the system in the terminal mode; see section 3.5.1.
2. Enter the family code and pinout code, if prompted by the terminal.

*NOTE*

*If options are desired (see section 3.5), select options and parameters as needed before proceeding.*

3. 


Terminal Displays



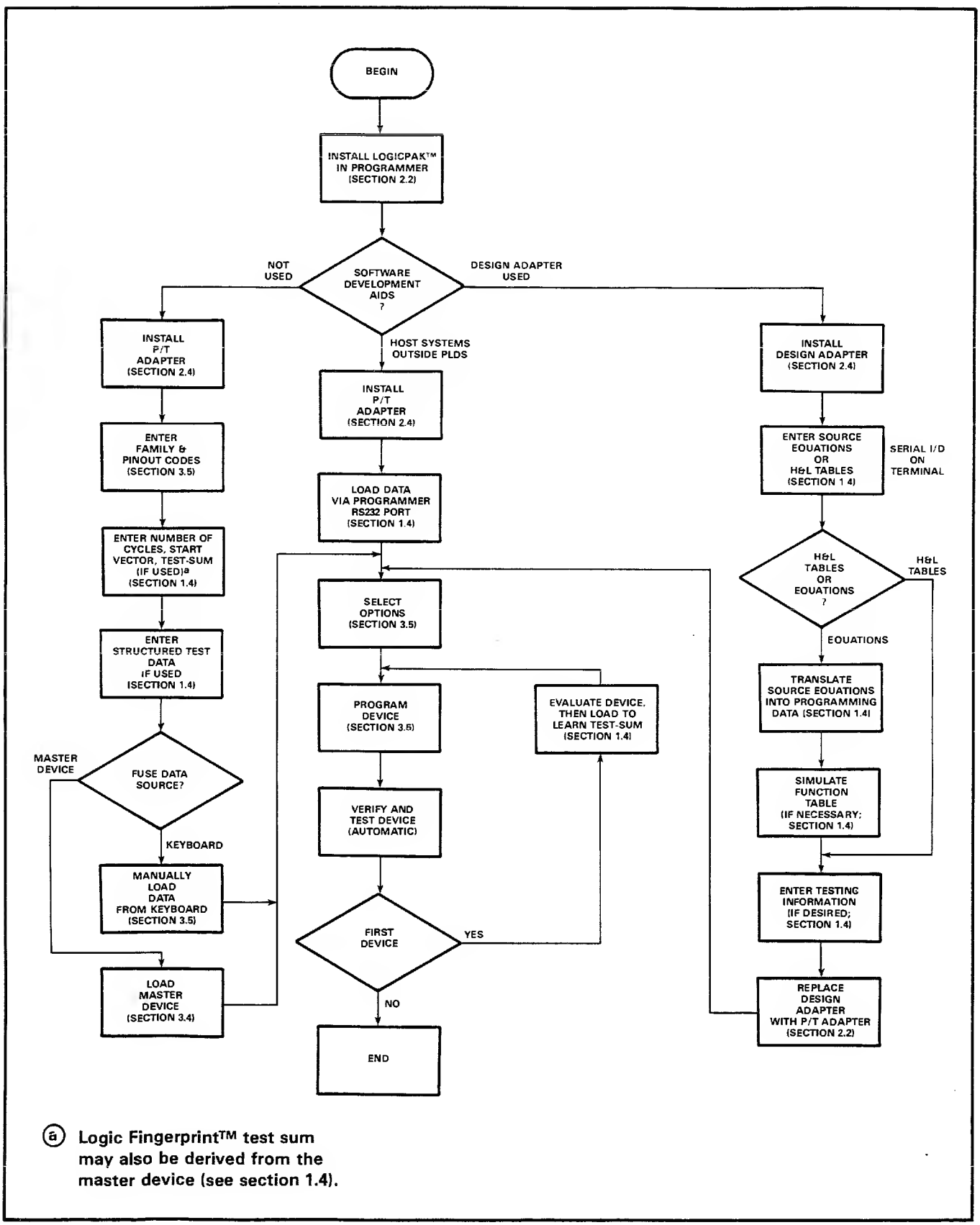

An action symbol will be displayed showing the verification function underway. Upon completion the terminal will display sum-check XXXX of the device fuses.

## 3.5 SYSTEM COMMANDS

In addition to the copy (load or program), verify, edit, and select functions described in the Operation Section of your programmer manual, the LogicPak™ offers numerous system commands that allow you to manipulate data and set parameters. System commands are accessed by entering a two-character select code from the programmer front panel or a one-character menu code from the terminal. Some commands will prompt for data entry. The operational overview (figure 3-5) will help you develop data and program a device using the system commands and programmer operations. Table 3-1 lists the select codes for Data I/O programmers to enter system commands from the programmer front panel and the menu codes for control from a terminal in terminal mode.

*NOTE*

*The sequence explanations assume no operating errors. If these occur, the programmer signals with a beep and displays a two-digit error code in front panel mode or an error message in terminal mode. It also beeps once when an incorrect key is pressed. Error codes are explained in section 4.1 (table 4-1) and in your programmer manual. Some errors will return you to the programmer front panel control from the terminal mode.*




ⓐ Logic Fingerprint™ test sum may also be derived from the master device (see section 1.4).

**Figure 3-5. Operational Overview Flowchart**

Table 3-1. PLDS System Command Summary

| MODULE OR ADAPTER | COMMAND TYPE | FROM FRONT PANEL | VIA TERMINAL | COMMAND DESCRIPTION | | SEE SECTION |
|---|---|---|---|---|---|---|
| LogicPak™ (with any adapter) | | -- | Ø | Display menu | | 3.5.2 |
| | | E 1 | -- | Enable terminal mode | | 3.5.1 |
| | | -- | 1 | Enter family code and pinout code | | 3.5.3 |
| | | E 5[a] | 5[a] | Enter reject count option | | 3.5.4 |
| | | E 6 | 6 | Enter verify option | | 3.5.5 |
| | | E 7 | 7 | Enter security fuse option | | 3.5.6 |
| | | E 8 | 8 | Set number of Logic Fingerprint™ test cycles | | 3.5.7 |
| | | E 9 | 8 | Enter starting vector and test-sum | | 3.5.7 |
| | | -- | 8 | Enter structured test vectors | | 3.5.7 |
| | | | | Ø | Display menu | 3.5.7 |
| | | | | D | Delete current vector | 3.5.7 |
| | | | | R | Repeat current vector | 3.5.7 |
| | | | | U | Display previous vector | 3.5.7 |
| | | | | #(N) | Go to vector (N) | 3.5.7 |
| | | | | space | Move cursor right | 3.5.7 |
| | | | | backspace[b] | Move cursor left | 3.5.7 |
| | | | | return | Display next vector | 3.5.7 |
| | | | | CTRL Z | Exit vector editor | 3.5.7 |
| | | E A | A | Display fuse pattern | | 3.5.8 |
| | | E B | B | Receive JEDEC data | | 3.5.9 |
| | | E C | C | Transmit JEDEC data | | 3.5.9 |
| | | E D | D | Display sum-check of fuse data | | 3.5.9 |
| | | E E | -- | Edit fuse by number | | 3.5.10 |
| | | -- | E | Edit fuse pattern | | 3.5.10 |
| | | | | Ø | Display menu | 3.5.10 |
| | | | | #(N) | Go to fuse (N) | 3.5.10 |
| | | | | space | Move cursor right | 3.5.10 |
| | | | | backspace[b] | Move cursor left | 3.5.10 |
| | | | | return | Display next row | 3.5.10 |
| | | | | CTRL Z | Exit fuse editor | 3.5.10 |
| | | E F | F | Display configuration number | | 3.5.11 |
| | | C E | G | Set option attributes | | 3.5.12 |
| | | -- | ESC | Exit terminal control before removing adapter | | 3.5.13 |
| [a]Except with PALASM adapter. [b]CTRL H is the same as backspace. | | | | | | |
| PALASM Design Adapter | Development | -- | Ø | Display menu | | |
| | | -- | 1 | Enter family and pinout codes | | |
| | | E 2 | 2 | Receive PALASM source | | |
| | | E 3 | 3 | Transmit PALASM source | | |
| | | E 4 | 4 | Assemble PALASM source | | |
| | | E 5 | 5 | Simulate function table | | |
| | Edit | -- | 9 | Edit source | | |
| | | -- | | Ø | Display menu | Refer to |
| | | -- | | B | Display line 1 | PALASM |
| | | -- | | C | Change text | Design |
| | | -- | | D | Delete character | Adapter |
| | | -- | | E | Display to end | Manual |
| | | -- | | I | Insert/enter text | |
| | | -- | | K | Delete current line | |
| | | -- | | L | Display 24 lines | |
| | | -- | | R(M)(N) | Repeat M lines after N | |
| | | -- | | U | Display previous line | |
| | | -- | | # ( N ) | Go to line N | |
| | | -- | | space bar | Move cursor/prompt right | |
| | | -- | | back space[b] | Move cursor/prompt left | |
| | | -- | | return | Display next line | |
| | | -- | | DEL/RUB | Delete characters (I mode) | |
| | | -- | | CTRL P | Purge all text | |
| | | -- | | CTRL Z | Exit editor, C or I mode | |
| | | -- | ESC | Exit terminal control before removing adapter | | |
| [b]CTRL H is the same as backspace. | | | | | | |

NOTE: ESC (escape) returns control to programmer front panel.

Table 3-1. PLDS System Command Summary (Cont.)

| MODULE OR ADAPTER | COMMAND TYPE | FROM FRONT PANEL | VIA TERMINAL | COMMAND DESCRIPTION | SEE SECTION |
|---|---|---|---|---|---|
| H&L Design Adapter | Development | -- | Ø | Display menu | Refer to H&L Design Adapter Manual |
| | | -- | 1 | Enter family and pinout codes | |
| | | E 2 | 2 | Receive data (IFL format)[c] | |
| | | E 3 | 3 | Transmit data (IFL format) | |
| | Edit | -- | 4 | Edit mode | |
| | | -- | | G Enter gate number | |
| | | -- | | P Enter product term number | |
| | | -- | | T Enter transition term number | |
| | | -- | | V Move cursor forward | |
| | | -- | | V Move cursor backward | |
| | | -- | | F Display next term | |
| | | -- | | R Display last term | |
| | | -- | | N Enter next field | |
| | | -- | | I Insert term | |
| | | -- | | D Delete term | |
| | | -- | | C Clear term | |
| | | -- | | X Deactivate term | |
| | | -- | | E Display edit sub-menu | |
| | | -- | | Ø Exit edit mode | |
| | | -- | | 1 Return to edit mode | |
| | | -- | | 2 Serial input (receive IFL format)[c] | |
| | | -- | | 3 Serial output (transmit IFL format) | |
| | | -- | | 4 List low-order terms | |
| | | -- | | 5 List high-order terms | |
| | | -- | | 6 Compress terms | |
| [c]Integrated fuse logic, Signetics ASCII | | -- | | CTRL Z Exit edit mode | |
| | | -- | ESC | Exit terminal control before removing adapter | |
| All P/T Adapters | Device | -- | 1 | Enter family code and pinout code | 3.4.1 |
| | | Load | 2 | Load fuse data from device to RAM | 3.4.4 |
| | | Verify | 3 | Verify fuse data and perform functional test | 3.4.6 |
| | | Program | 4 | Program device with RAM data | 3.4.5 |

NOTE: ESC (escape) returns control to programmer front panel

### 3.5.1 ENABLE TERMINAL MODE


Select code E1 transfers control of the PLDS to the terminal. After control is transferred, the 29A will display only its action symbol. This command allows you to access data development and remote operations resident in the design adapters and remote operations using the P/T adapters.

The terminal will prompt you to enter family codes and pinout codes unless they have already been entered. For P/T adapters only; see section 3.5.3. The terminal will then display the command menu (see figure 3-6).

See section 2.6 for terminal setup procedure.

### 3.5.2 DISPLAY COMMAND MENU


This command causes the PLDS to redisplay its command menu on the terminal as shown in figure 3-6.

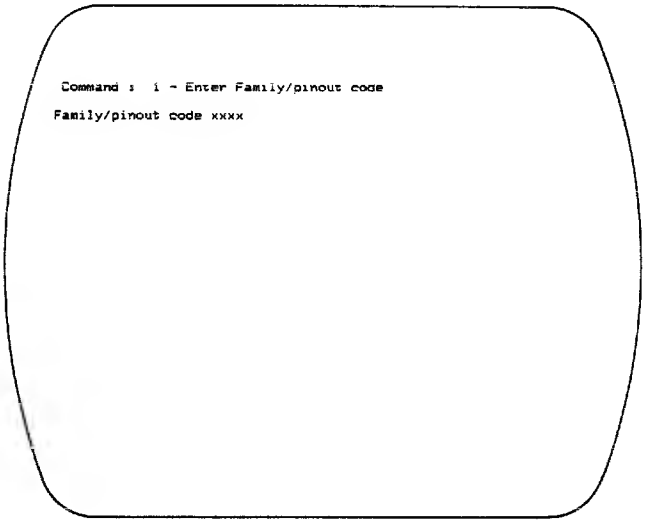
```
 Command :  0 - Display menu


DATA I/O CORP. - Programmable Logic Development System - 303A-V02
                        Copyright 1982,1983

      - GENERAL COMMANDS -                    - I/O COMMANDS -
0 - Display menu                          B - Receive JEDEC data
1 - Enter Family/pinout code              C - Transmit JEDEC data
5 - Enter reject count option
6 - Enter verify option
7 - Enter security fuse option
8 - Enter functional test data
F - Configuration number
G - Select attributes

    - DEVICE RELATED  COMMANDS -              - FUSE MAP COMMANDS -
2 - Load device                           A - Display fuse pattern
3 - Verify device                         D - Display fuse sumcheck
4 - Program device                        E - Edit fuse pattern

NOTE - Always transmit an "ESC" before removing adapter
```

**Figure 3-6. PLDS Command Menu**

### 3.5.3 FAMILY CODE AND PINOUT CODE

From the 29A front panel control, family code and pinout code entry is part of device-related operations (see sections 3.4.1 through 3.4.4).


Terminal Displays




Enter the family code and pinout code (see section 3.4.1 for more detail). Space and backspace (CTRL H) may be used to move the cursor back and back and forth.

### 3.5.4 SET REJECT COUNT OPTION

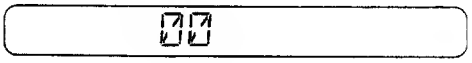
This command allows you to select the number of programming pulses applied to the device fuses before the programmer rejects the device as unprogrammable. The default value of 0 selects the manufacturer's specified number of programming pulses. Refer to the adapter manual for specific entries to select optional reject values for single-pulse, military specifications, etc.

*NOTE*

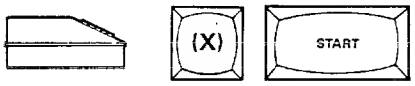
*The PALASM adapter does not provide this option.*

Front Panel Operation

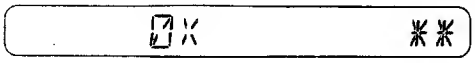

29A Displays




To change the reject count to an optional value, enter the code number (X) specified in the adapter manual.

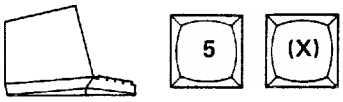



29A Displays

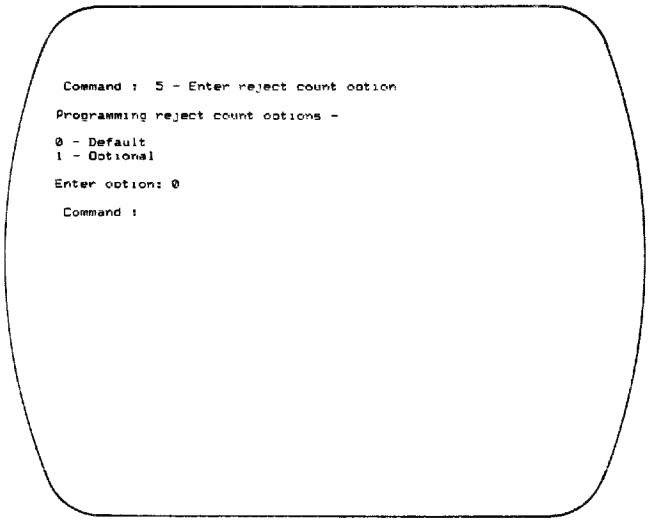


### Terminal Operation

To select the reject count from the terminal enter a 5 from the command mode, then respond to the prompt with the count (X).




Terminal Displays




### 3.5.5 SELECT VERIFY OPTION

Three options are available for selecting verify and functional test routines. These routines are described in detail in sections 3.4.6 and 1.4.3.

Options available are:

| OPTIONS | DESCRIPTION |
|---|---|
|  | Default option. Perform fuse verify, followed by structured test (if test vectors are present in RAM), and Logic Fingerprint™ test (if one or more Logic Fingerprint™ test cycles are selected), in that order. |
|  | Perform fuse verify only. |
|  | Perform structured test and Logic Fingerprint™ test only, in that order. Do not perform fuse verify. |

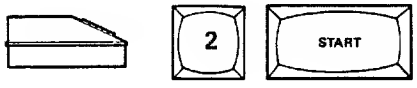
Option 0 (default) is the option used in normal operation. Option 1 checks the programming of the device fuses without checking device functionality. Use option 2 to functionally test devices with the security fuse blown. In addition, option 2 can be used to learn the Logic Fingerprint™ test of a device with the security fuse blown. Fuse data in RAM will be cleared during this operation. Programming can not occur with option 2 selected.

Verify options must be entered from the programmer's keyboard or a terminal. The option will remain in effect until it is changed or until the unit is powered down. To reselect the default, key in option 0.

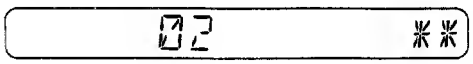
### Front Panel Operation


29A Displays


At this point, to select functional test **only** for example, do the steps which follow.

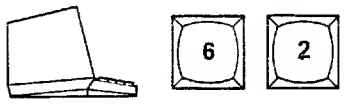



29A Displays



### Terminal Operation

To enter the verify option from the terminal, enter 6 from the command mode, then respond to the prompt with the desired option. For example, to select functional test only:




Terminal Displays

```
Command : 6 - Enter verify option

0 - Sequence - fuse verify, structured test, Logic Fingerprint
1 - Fuse verify only
2 - Sequence - structured test, Logic Fingerprint

Enter verify option: 2

Command :
```

### 3.5.6 SELECT SECURITY FUSE OPTION

Some logic devices are equipped with protective fuses called security fuses. Once the security fuses are programmed, the fuse states in the logic array cannot be copied. Programming the security fuses makes it very difficult to pirate a device design.

The PLDS security fuse programming feature is a fail-safe function. You can either enable programming of the security fuse at all times, only allow programming when security fuse data are downloaded to the PLDS via the serial port, or disable programming completely, whether security fuse data are downloaded or not.

When the security fuse has been blown, a Logic Fingerprint™ test and structured test can still be performed, but a fuse verify operation is not possible (see section 3.5.5).

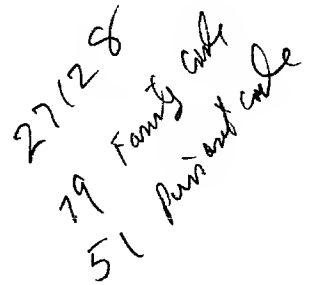
To enable programming of security fuses two conditions must be met: 1) the security fuse state in the programmer RAM must be 1 (or true), and 2) security fuse programming must be enabled. Once the security fuse option is selected, it will remain in effect until changed or until the programmer is turned off.

When security fuse data are entered into RAM in the JEDEC ASCII-logic format (see section 1.4.1), data in the G field indicate the state of the security fuse. The G field does not affect the enable state of the security fuse option; the enable state must be entered separately. This can be done before or after loading JEDEC ASCII-logic format data.

Security fuse states cannot be loaded from a master device.

**CAUTION**

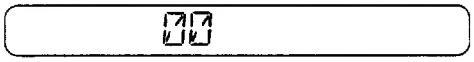
**Once the security fuse is blown, you can no longer verify the state of any fuse in the device. The process cannot be reversed; therefore, be certain that you want to program the security fuse before you activate this function. Attempting to re-program the device after the security fuse is blown will alter the original fuse pattern and render the device inoperative.**



### Front Panel Operation

To select a security fuse option from the front panel:


29A Displays

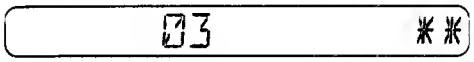


Security fuse select-code options are:

| OPTION | DESCRIPTION |
|---|---|
| 0 | Default option. Disable programming and set the security fuse state in RAM to 0 (unprogrammed). |
| 1 | Disable programming, and set security fuse state in RAM to 1 (programmed). |
| 2 | Enable programming, and set security fuse state in RAM to 0. (Data downloaded in the JEDEC format can change the security fuse state to 1.) |
| 3 | Enable programming, and set security fuse state in RAM to 1. (Data downloaded in the JEDEC format can change the security fuse bit back to 0.) |

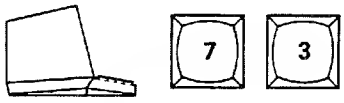
For example, to enable security fuse programming and set security fuse state in RAM to 1 (option 3):


29A Displays

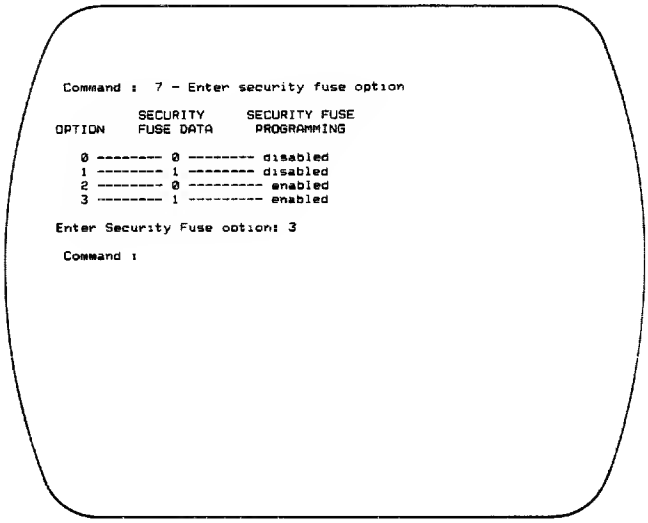


### Terminal Operation

To enter the security fuse option from the terminal, enter 7 from the command mode, then respond to the prompt with the desired option. For example, to enable security fuse programming and set security fuse state in RAM to 1, do the following:



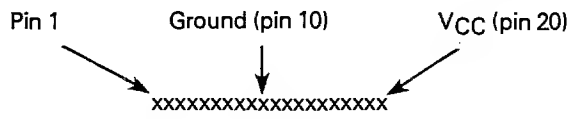

Terminal Displays



### 3.5.7 ENTER FUNCTIONAL TEST DATA

Functional test data includes information for the Logic Fingerprint™ test and also the test vectors used by P/T adapters for testing of a programmed device. The Logic Fingerprint™ test information consists of three components:

- The number of test cycles to be performed during the Logic Fingerprint™ test (described later in this subsection). The default value is 00, which disables the Logic Fingerprint™ test.
- The Logic Fingerprint™ starting vector. This is an arbitrary binary sequence, each bit of which corresponds to a pin on the device under test. The starting vector format for a 20-pin device is shown below. Each "x" represents a "1" or a "0" to apply a logic high or logic low to the corresponding pin. Values entered for $V_{CC}$ and ground affect the resulting Logic Fingerprint™ test signature, but have no effect on the device under test.




  The starting vector is used to initialize the Logic Fingerprint™, and is one of the components (along with the device type, number of test cycles, and programming pattern) which determines the resulting Logic Fingerprint™ test signature. Note that different Logic Fingerprint™ test signatures may result for a given logic design, depending on the choice of starting vector.
- The Logic Fingerprint™ test signature itself is the result of performing the Logic Fingerprint™ test as described later in this subsection.

Logic Fingerprint™ test data may be entered either from the front panel or from the terminal. From the front panel, the number of test cycles and starting vector for the Logic Fingerprint™ test may be entered, and the resulting Logic Fingerprint™ test signature may be viewed or entered. Structured test vectors may not be entered or edited from the front panel but only from a terminal or serial download. All functional test data may be entered from the terminal, including number of test cycles, starting vector, the Logic Fingerprint™ test signature itself, and the test vectors.

*NOTE*

*If a value is entered for the Logic Fingerprint™ test signature, it should be either 00000000 or a known-good value corresponding to the number of test cycles, starting vector, device, and fuse patterns under test. A value of 00000000 will cause the LogicPak™ to "learn" the correct Logic Fingerprint™ test signature when a Load, Program, or Verify operation is performed (see section 3.4 for details). When in Load, the correct Logic Fingerprint™ will be learned independent of the value entered.*

*If "Device Selection Error" (Prog Pak 30) appears when you select functional test data, you must specify family code and pinout code to define the vector width.*

In the subsections which follow, functional test data will be entered to test the Basic Gates design example[a] (see figure 3-10).

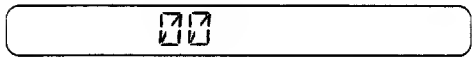
[a]Adapted from THE MMI PAL HANDBOOK, available from Monolithic Memories Inc., 1165 Arques Avenue, Sunnyvale California 94086.

### Front Panel Operation

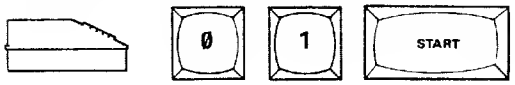
From the front panel, the number of test cycles and the Logic Fingerprint™ starting vector may be entered, and the Logic Fingerprint™ test signature may be viewed or entered.

**Set Number of Logic Fingerprint™ Test Cycles.**

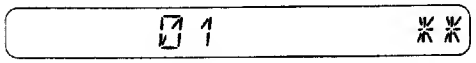

29A Displays



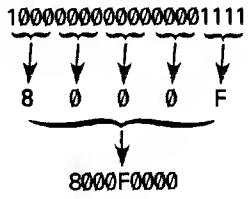

For example, to enable one cycle of testing,




29A Displays




**Logic Fingerprint™ and Starting Vector.** The starting vector must be converted from the binary form to hexadecimal for entry from the front panel. For our Basic Gates example, we will choose an arbitrary test vector as shown:




Starting vector (binary)

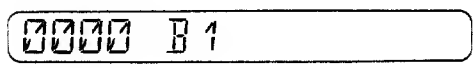
(hexadecimal)

Starting vector (hexadecimal)

The unused portion of the 32-bit vector is assumed to be zeroes and must be included in the hexadecimal vector entry.

For example:

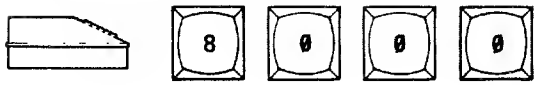

29A Displays



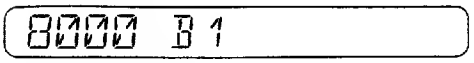

*NOTE*

*The eight-character starting vector is entered into the programmer in two fields. B1 identifies the first field.*

To enter the first four hexadecimal digits,

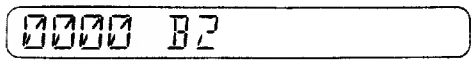



29A Displays





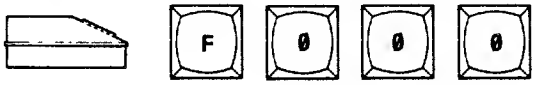
29A Displays



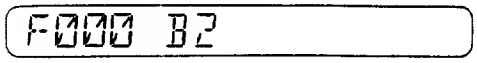

*NOTE*

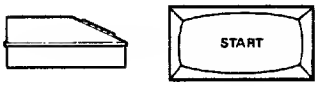
*B2 represents the second field.*

Enter the remaining hexadecimal digit by pressing




29A Displays



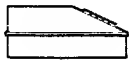
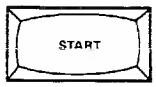




The zeroes are ignored, but are needed to correctly position the "F."

This vector, when applied to the Basic Gates example, produces the Logic Fingerprint™ test signature:

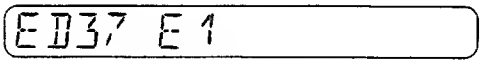
ED37A9E4 (hexadecimal)

This value may be viewed or entered at this time by pressing START:

1.  

Enter the first four characters of the Logic Fingerprint™ test signature if desired.

29A Displays

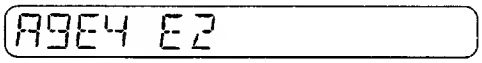


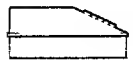
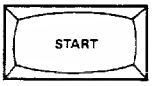

The first four characters are displayed as the E1 field. The last four characters (E2 field) may be viewed or entered by pressing START again:

2.  

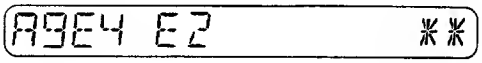
Enter the next four characters if desired.

29A Displays



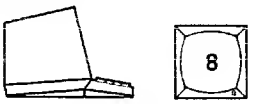

3.  


29A Displays




### Terminal Operation

Entering an "8" from the Command mode allows you to enter functional test data and begin vector editing from the terminal.




The functional test data may be entered in response to three prompts (see figure 3-7).

```
Command : 8 - Enter functional test data

Cycles for Fingerprint: xx
Fingerprint starting vector: xxxxxxxxxxxxxxxxxx
Fingerprint: xxxxxxxx

Note: x.represents current values
```

**Figure 3-7. Prompts for Entering Functional Test Data**

As each prompt appears, you may modify the current values (represented by x's in figure 3-7) using the following steps:

1. Move the cursor forward (using the spacebar) and backward (using the backspace) along the displayed value until it is positioned over the symbol to be changed.
2. Type the desired symbol.
3. Enter RETURN or CTRL Z at any point to move to the next prompt.
4. CTRL Z is used to exit the functional test entry mode.

For our test example, the values shown in figure 3-8 should be entered.

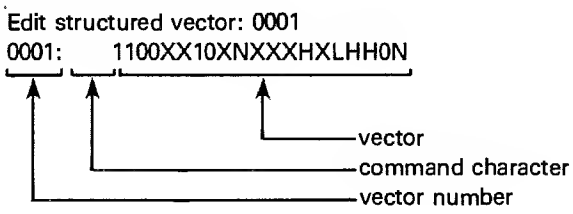
```
Command : 8 - Enter functional test data
Cycles for Fingerprint: 01
Fingerprint starting vector: 10000000000000001111
Fingerprint: ED37A9E4

          - DISPLAY -                        - EDITING COMMANDS -
0 ------------ Display menu            D ------- Delete (Kill) current vector
Return -------- Go to next vector       R ------- Repeat current vector
U ------------- Up (previous vector)    CTRL Z -- Exit vector editor
#(N) ---------- Go to vector (N)
Space --------- Move cursor right
BKSP (CTRL H) - Move cursor left
CTRL Z -------- Exit vector editor

Edit structured vector:  0000
0001:  1100XX10XNXXXHXLHH0N
0002:  --------------------
```

**Figure 3-8. Entering Functional Test Data**

**Vector Editing.** Vectors are created by downloading JEDEC 'V' fields, simulating a source file containing a function table, or by using the vector editor.

When the Logic Fingerprint™ test information has been entered (or skipped by entering RETURNs), the vector editor menu appears (see figure 3-8), and a prompt appears for the vector number to be edited. The default vector is 0001, as shown in figure 3-8.

The vector editor is a fixed-format line editor with the first column of the displayed line reserved for command characters as shown below.




A character entered in the first column (normally blank) is interpreted as a command and acted upon immediately; otherwise vector editing is not processed until a RETURN is entered (at any point on the line). The command characters recognized in the first column are 0, U, #, K, and R; see table 3-2 for command character definitions.

During operation, the vector editor copies the selected vector to a temporary buffer where all editing changes are made. Then, when a RETURN command is entered, the temporary buffer is examined for legal characters before copying back to vector memory. You are not allowed to proceed to another vector until all characters are legal in the current vector. Typing a CTRL Z to exit the vector editor will leave the selected vector in its original state.

**Table 3-2. Vector Editor Command Characters**

| COMMAND | DESCRIPTION | ACTIVITY |
|---|---|---|
| 0 (zero) | Display menu | Redisplays menu and restarts editing on the same vector. |
| U | Up (previous vector) | Moves editing to the next lower vector number (the vector one 'up' on the screen). |
| #(N) | Go to vector (N) | Entering a '#' in the command column causes the vector editor to prompt for the desired vector number (default = 0001). Entering a vector number greater than the last vector will move you to the last vector. |
| D | Delete current vector | Current vector is deleted, and all higher vectors moved down one. Current vector number is redisplayed with new vector. |
| R | Repeat current vector | Creates a copy of the current vector immediately following the current vector. The copy is displayed, with its vector number (one greater than the original). This command may be given for any vector, and existing vectors will be moved to accommodate the new copy. |

An "empty vector" is represented by a dash in all pin positions. This will appear as the first vector in an empty vector editor buffer, or as one past the last vector where data are present in memory. All vectors are numbered lower than the empty vector.

To edit a vector, follow the steps below.

1. Move the cursor forward (using the spacebar) and backward (using the backspace) along the displayed vector until it is positioned over the test condition to be changed.
2. Type the desired test condition to enter it into the vector image; the allowable test conditions are 0-9, X, N, F, H, L, Z, C, and K (see table 3-3 for test condition definition).

Table 3-3. Vector Symbol Definition

| VECTOR SYMBOL | DEFINITION |
|---|---|
| 0 | Drive input low |
| 1 | Drive input high |
| 2-9 | Drive input to supervoltage #2-9 |
| C | Drive input low, high, low |
| K | Drive input high, low, high |
| N | Power pins and outputs not tested |
| L | Test output low |
| H | Test output high |
| Z | Test output for high impedance |
| F | Float input or output |
| X | Ignore input or output (not defined in JEDEC format) |

*NOTE*

*"X" is not defined in the JEDEC format. The "X" is treated as an "N" for outputs and leaves an input at its previously defined state.*

Test conditions 2 through 9 specify non-TTL levels (super voltages) that access special device features. A device may be damaged by improper use of super voltages.

3. Enter RETURN or CTRL Z at any point to move to the next vector or to exit the vector editor.

### 3.5.8 DISPLAY FUSE PATTERN

This command transmits the fuse pattern in the programmer data RAM to the serial port. The fuse states may be shown as a series of "1"s and "0"s or a series of "-"s and "X"s; see section 3.5.12 on selecting characters. The "1"; or "-" represents a high resistance fuse, "blown" in a fuse link device. The "0" or "X" represents a low resistance or "intact" fuse. Each fuse can be identified by a decimal fuse number as shown in figure 3-9. The fuse states are arranged in a matrix that corresponds to the logic diagram of the device (figure 3-10). This is useful for comparing or copying a displayed fuse pattern to the device logic diagram. Logic diagrams and fuse number charts for all supported devices are in the appendix of the programming/testing adapter manuals.

```
Command : A - Display fuse pattern

      00          10          20
0000  -------X--  ----------  ----
0024  XXXXXXXXXX  XXXXXXXXXX  XXXX
0048  XXXXXXXXXX  XXXXXXXXXX  XXXX
0072  XXXXXXXXXX  XXXXXXXXXX  XXXX
0096  X-X-------  ----------  ----
0120  XXXXXXXXXX  XXXXXXXXXX  XXXX
0144  ----X-----  ----------  ----
0168  --------X-  ----------  ----
0192  ----------  -X-X------  ----
0216  XXXXXXXXXX  XXXXXXXXXX  XXXX
0240  ----------  -----X--X--  ----
0264  ----------  -----XX---  ----
0288  ----------  ----------  -X--
0312  ----------  ----------  ---X
0336  ----------  ---------X  ----
0360  XXXXXXXXXX  XXXXXXXXXX  XXXX
Sumcheck 1BB9

Command :

NOTE: - = open
      X = intact

Fuse Number = First Fuse Number + Increment
```

**Figure 3-9. Complete Fuse Pattern**

*NOTE*

*Sending certain control characters to the PLDS during the course of fuse pattern display will affect the display. The output may be stopped by sending a CONTROL S (DC1 or ASCII 11 hex) and then restarted by sending a CONTROL Q (DC3 or ASCII 13 hex).*

*A CONTROL Y (ASCII 19 hex) will terminate the transmission and return to the terminal or front panel operation.*

*An ESCAPE character (ASCII 1B hex) will terminate the transmission and return to front panel operation.*

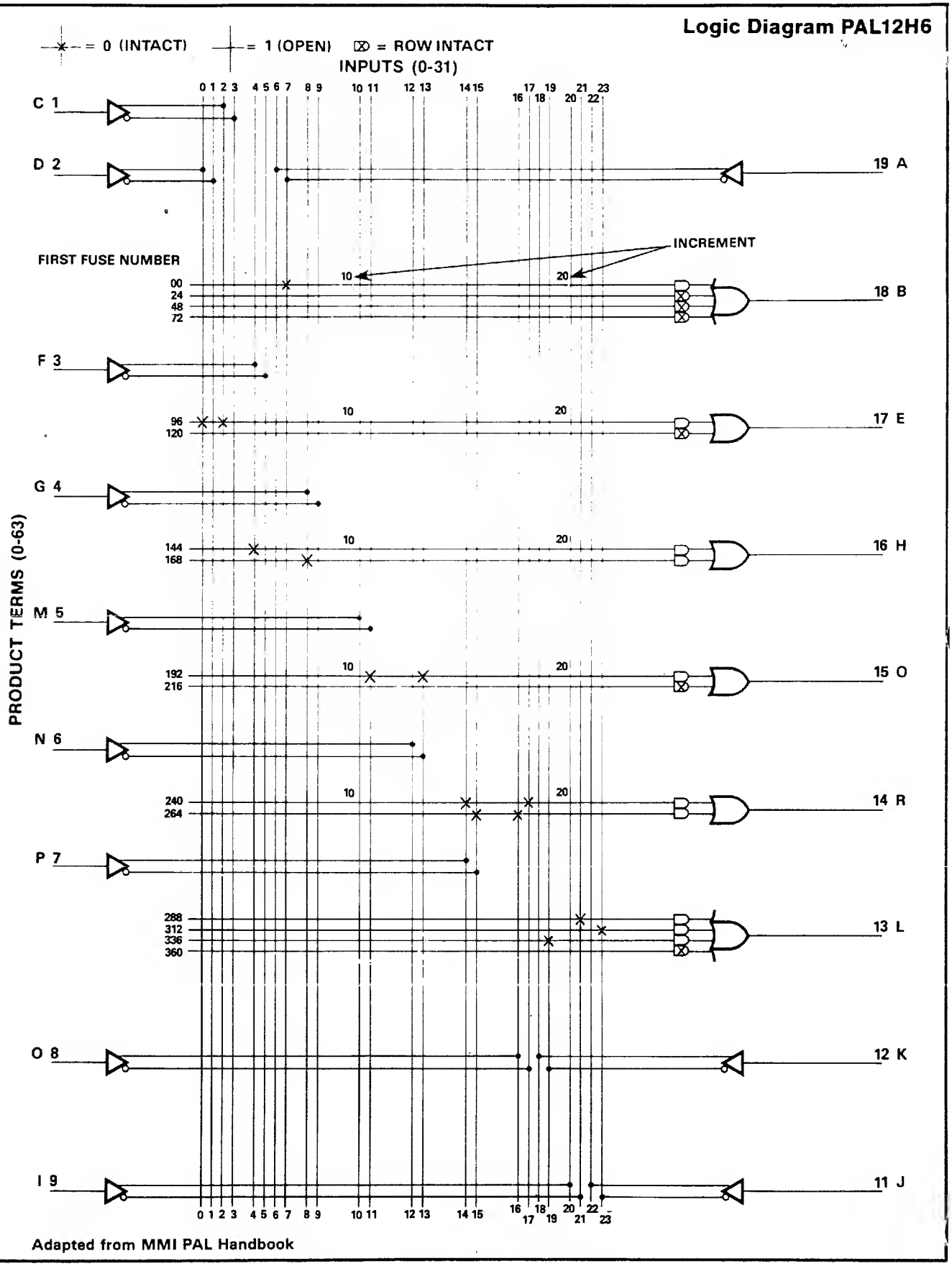
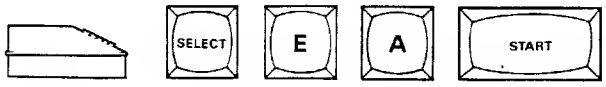



Figure 3-10. Logic Diagram for Basic Gates Example

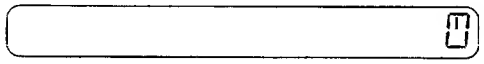
The last character of the fuse pattern transmission is either CONTROL C (ETX or ASCII 03) or a CONTROL Z (ASCII 1A hex). (See section 3.5.9 on selecting the termination character.)

### Front Panel Control

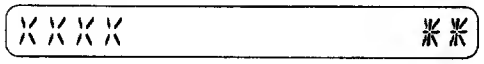
To display the fuse pattern from front panel control, follow these steps:




29A Displays



29A Displays



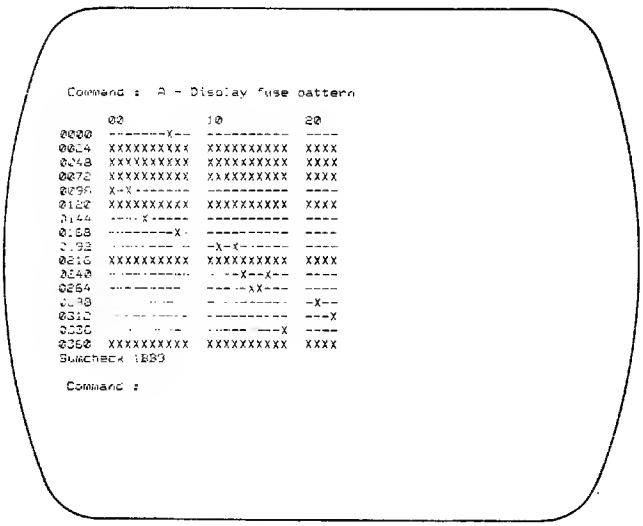
*NOTE*

*[1] is the action symbol. XXXX is the fuse array checksum.*

### Terminal Control

To display the fuse pattern from the terminal command mode, enter an "A":


Terminal Displays




### 3.5.9 JEDEC FORMAT DATA EXCHANGE

Fuse data, test vectors, and the Logic Fingerprint™ test signature are transmitted between the host computer and the PLDS in the JEDEC format. The JEDEC format is described in detail in appendix A. A brief overview of the format is provided in this section, and shown in figure 3-12. Figure 3-11 shows an example JEDEC transmission and its components.

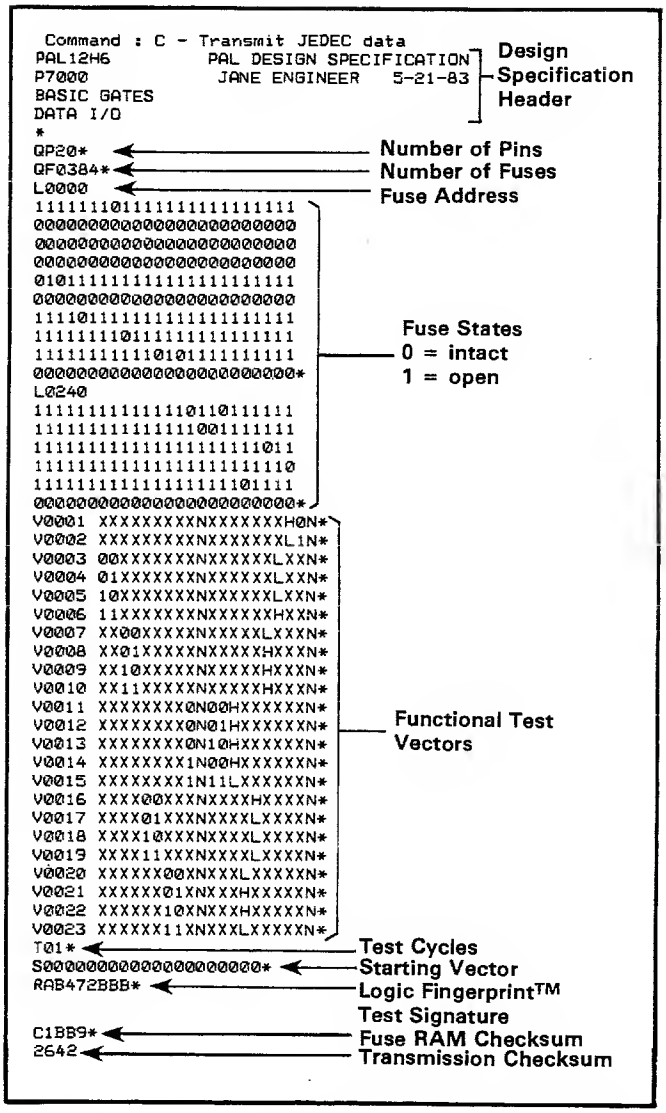



**Figure 3-11. JEDEC Transmission—Basic Gates Example**

The transmission consists of a start-of-text (STX) character, the various fields, an end-of-text (ETX) character, and a transmission checksum as shown in figure 3-12.

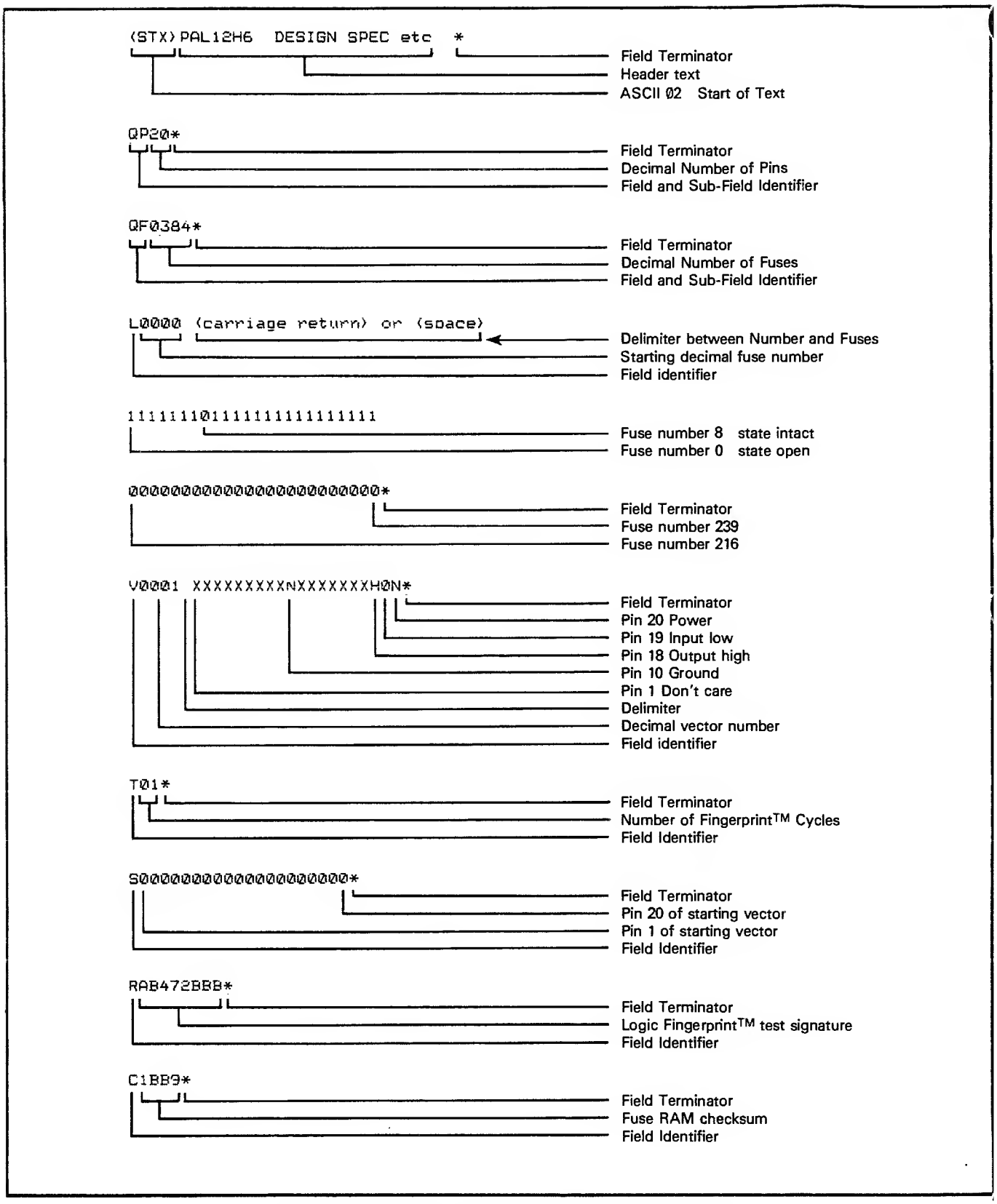



**Figure 3-12. JEDEC Format (Breakdown of Figure 3-11)**

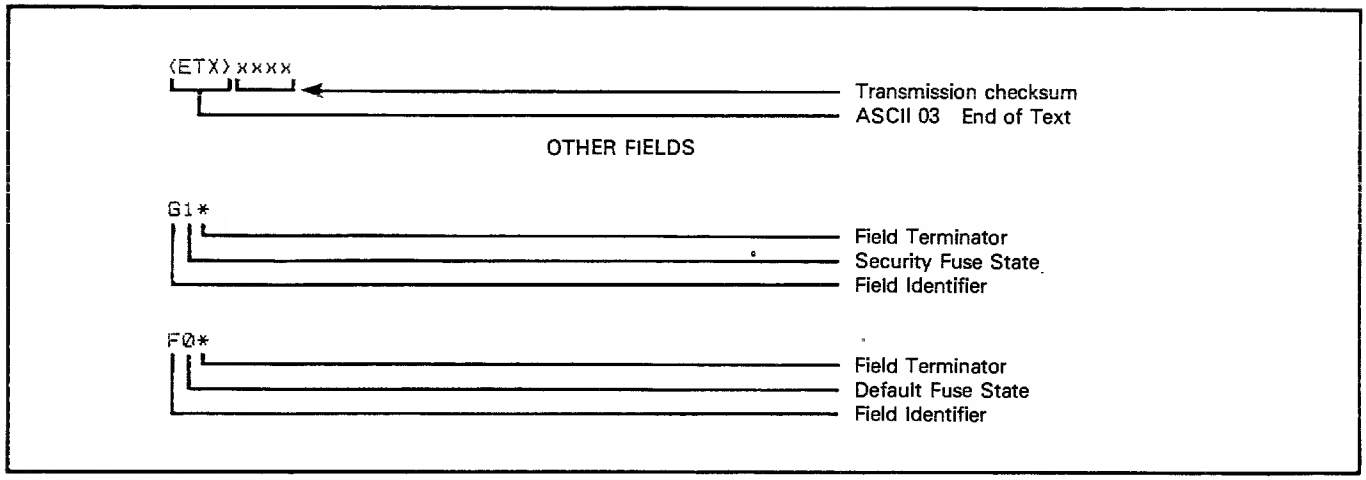



**Figure 3-12. JEDEC Format (Breakdown of Figure 3-11) (Cont.)**

The transmission checksum is the 16-bit sum of all ASCII characters transmitted between and including the STX and ETX. The parity bit is excluded in the calculation (see figure 3-13). The transmission checksum computed by the PLDS may be found by examining data RAM addresses 405 and 406, using the programmer's EDIT mode (discussed later in this subsection). Some computer operating systems do not allow a user to control what characters are sent, especially at the end of a line. The transmission checksum may be disabled in this case by sending a dummy checksum of "0000".

In general each field in the format starts with an identifier, is followed by the information, and is terminated with an asterisk. For example, "T01*" sets the number of Logic Fingerprint™ test cycles to 1. The design specification header does not have an identifier and must be the first field in the transmission, immediately following the STX.

Fuse information is specified by the "QF", "F", "L", and "C" fields. The "QF", "F", and "C" fields are optional.

The "QF" field sets the maximum allowable fuse number; this will override the value set by the family code and pinout code. The "F" field sets the default fuse value. An "F0*" fills the fuse RAM with 0s, and an "F1*" fills the fuse RAM with 1s. This operation takes a significant amount of time and can lead to an input buffer overflow at high baud rates.

The "L" field starts with a decimal fuse number and is followed by a stream of fuse states (1 or 0). The fuse number may include leading zeroes (i.e. "L12" and "L0012" are the same). A "space" and/or a "carriage return" must separate the fuse number from the fuse states. The stream of fuse states can be as long as desired (up to the maximum allowable fuse number). The fuse data for an entire device, for example, could be sent in one "L" field starting at zero and continuing for all fuses in the device. Spaces and carriage returns may be inserted to make the stream more readable. Each "L" field must be terminated with an asterisk.

```
(STX)*(ETX)002F          02 + 2A + 03 = 2F


random text (return)(line feed)                                    = 0000
(STX)TEST*(return)(line feed)          02+54+45+53+54+2A+0D+0A     = 0183
QF0384*(return)(line feed)          51+46+30+33+38+34+2A+0D+0A     = 01A7
F0*  (return)(line feed)                  46+30+2A+20+20+0D+0A     = 00F7
L10 101*(return)(line feed)   4C+31+30+20+31+30+31+2A+0D+0A     = 01A0
(ETX)05C4 (return) other random text                          03 = 0003
                                                                   ----
                                                                   05C4
```

**Figure 3-13. Computing the Transmission Checksum**

The "C" field is the sum-check of the entire fuse RAM (fuse number Ø to maximum fuse number for the selected device), not just the fuse states sent. See figure 3-14. (The JEDEC term "Fuse Checksum" is the same as Data I/O's term "sum-check".)

The structured test vector information is specified by the "QP", "P", and "V" fields. The "QP" field defines the number of pins on the device and overrides the value set by the family code and pinout code. The "V" field starts with a vector number, is followed by a space, then by a series of test conditions for each pin, then is terminated with an asterisk. The test conditions are normally sent in pin number order, however the "P" field can specify a different sequence. The PLDS JEDEC translator does not validate the test conditions in the vectors (see table 3-3 for the presently defined test conditions). The super-voltage test conditions (2 through 9) are used to apply non-TTL levels to certain pins to access special test features. A device could be damaged by improper use of super-voltages.

The Logic Fingerprint™ Test information is specified by the "T", "S", and "R" fields. The "T" field defines the number of test cycles to be performed. The legal values are Ø to 99. The "S" field defines the starting vector with a series of 1s and Øs for each pin. The "R" field defines the 8 digit hexadecimal Logic Fingerprint™ test signature.

The "G" field defines the security fuse state.

The "D" field is not sent by new versions of the PLDS JEDEC translator. It has been replaced by the "QF" and "QP" fields and the manual setting of family codes and pinout codes.

```
(STX)*F0*L0000
01001110 00001000 11110000 11111111 01010001*
C021A*
(ETX)0000

The F0* cleared all the fuse RAM to 0. The L field transmitted 40
fuse states starting at 0.

Fuse number 00 01 02 03 04 05 06 07  08 09 10 11 12 13 14 15  16
State        0  1  0  0  1  1  1  0   0  0  0  0  1  0  0  0   1


                MSB          LSB

                 7 6 5 4 3 2 1 0

  0000          0 1 1 1 0 0 1 0          72
  0008          0 0 0 1 0 0 0 0          10
  0016          0 0 0 0 1 1 1 1          0F
  0024          1 1 1 1 1 1 1 1          FF
  0032          1 0 0 0 1 0 1 0          8A
  0040          0 0 0 0 0 0 0 0          00
  0048          0 0 0 0 0 0 0 0          00
  xxxx          0 0 0 0 0 0 0 0          00
                                        ----
                                        021A
```

**Figure 3-14. Computing the Fuse RAM Checksum**

### Transmit JEDEC Data

This command transmits the contents of the fuse and vector RAM to the serial port in the JEDEC format (see appendix A).

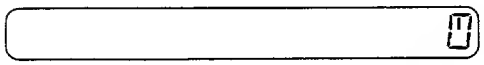
The following characteristics apply to JEDEC transmission.

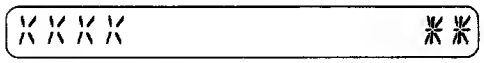
- The output may be halted by sending a CONTROL S (DC1 or ASCII 11 hex) and restarted by sending a CONTROL Q (DC3 or ASCII 13 hex).
- An ESCAPE character (ASCII 1B hex) will abort the transmission and return to the programmer front panel operation.
- A CONTROL Y (ASCII 19 hex) will terminate the transmission and return to the terminal or programmer front panel operation.
- The Logic Fingerprint™ test fields (S, R, and T) are not sent if the number of cycles is Ø.
- The "G" field is sent only if security fuse data is a "1."
- The fuse checksum (C field) is the 16-bit sum of all fuse states (*i.e.*, from fuse Ø to the fuse limit for the device). See figure 3-14.

### Front Panel Control

To transmit JEDEC data, follow the steps below.


29A Displays





*NOTE*

*□ is the action symbol. XXXX is the fuse array sum-check.*

### Terminal Control

To receive JEDEC data from the terminal mode, enter a "C" from the Command mode:


See figure 3-11 for the terminal display of the Basic Gates design example data.

### Receive JEDEC Data

This command prepares the programmer to receive fuse and vector data from a peripheral device via the serial port. A translator converts the JEDEC format data (see appendix A) to the memory image required by the PLDS.

*NOTE*

*The D field is ignored by the translator. The correct family code and pinout code must be entered before receiving JEDEC data. See table B-1 in appendix B for correct family codes and pinout codes.*

There are three types of errors that may be caused by receiving improper data in the JEDEC format (see table 3-4). You may determine the field in which the error

**Table 3-4. Translator Input Errors**

| ERROR | DESCRIPTION | POSSIBLE FIELDS |
|---|---|---|
| 82 | SUMCHK ERR | Transmission check-sum |
| 84 | INVALID DATA | ETX F L S V |
| 91 | I/O FORM ERR | C G L P R T V |

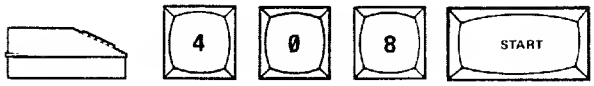
occurred by examining data RAM location 0408; the ASCII value (hexadecimal) of the field is stored here (see table 3-5). More information about the possible cause of the error may be found in table 3-6.

**Table 3-5. ASCII Values of Field Identifiers**

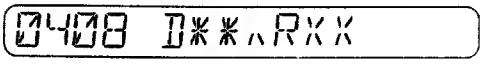
| FIELD IDENTIFIER | ASCII VALUE |
|---|---|
| (ETX) | 03 |
| C | 43 |
| F | 46 |
| G | 47 |
| L | 4C |
| P | 50 |
| R | 52 |
| S | 53 |
| T | 54 |
| V | 56 |

To examine the data RAM location 0408, perform these steps.


29A Displays


29A Displays




*NOTE*
*XXXX is the current address.*
*XX is the field identifier in hexadecimal.*

The transmission checksum computed by the PLDS may be found by examining data RAM locations 405 and 406 in a similar fashion.

**Table 3-6. Translator Input Error Codes**

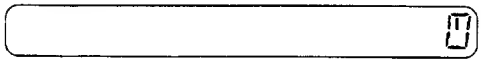
| ERROR | DISPLAY | FIELD* | POSSIBLE CAUSE |
|---|---|---|---|
| 82 | SUMCHK ERR | ETX | Transmission checksum of all ASCII characters does not match the computed value. |
| 84 | INVALID DATA | ETX | Fuse sum-check does not match computed sum-check. The comparison is not made until the transmission is complete, so the field is stored as ETX rather than C. The sum-check includes the entire fuse RAM as defined by the family and pinout code, not just the fuse states sent. |
| | | F | Invalid character in field. Only "1" and "0" are allowed. |
| | | L | A space or carriage return did not follow the fuse number. |
| | | L | An invalid character was in the fuse state field. Only "1" and "0" are allowed. Spaces, line feeds, and carriage returns are ignored. |
| | | S | Invalid character in field. Only "1", "0", and "N" are allowed. |
| | | V | Too few or too many test conditions. |
| 91 | I/O FORM ERROR | C | Invalid character in field, must be 4 digit hexadecimal number. |
| | | G | Invalid character in field. Only "1" or "0" are allowed. |
| | | L | Fuse number exceeds fuse limit for device or invalid fuse number (must be decimal number). |
| | | P | Too few or too many pins or invalid pin number for device. |
| | | T | Test cycles greater than 99. |
| | | R | Invalid character in field: must be 8 digit hexadecimal number. |

*From RAM Addr. 0408

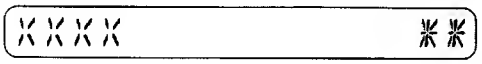
Front Panel Control

To receive JEDEC data from the front panel mode perform the steps which follow.


29A Displays



29A Displays

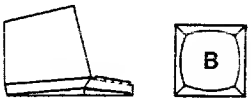


*NOTE*

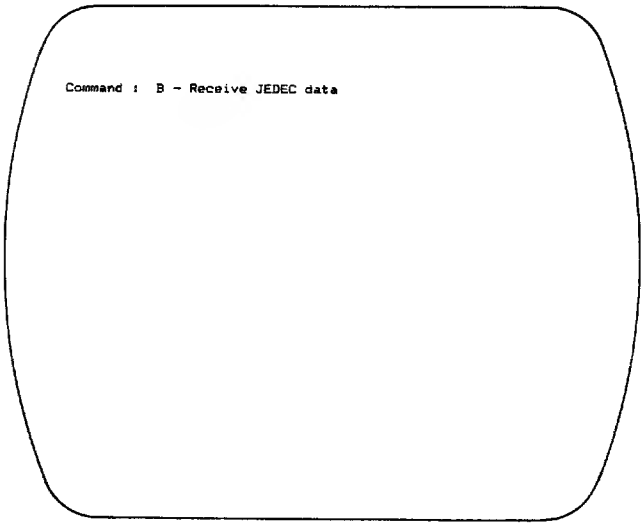
*[1] is the action symbol. XXXX is the fuse array sum-check.*

Terminal Control

To receive JEDEC data from the terminal mode, enter a "B" from the Command mode:




Terminal Displays




### 3.5.10 EDIT FUSE PATTERN

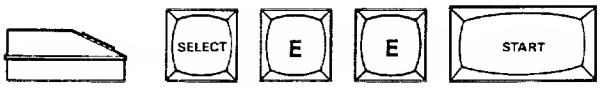
The individual fuses that make up a PAL® fuse map may be edited using the fuse map editor. Fuses may be changed from blown to unblown or vice-versa on a downloaded fuse map, a fuse map generated by assembly of source code, or directly in fuse memory.

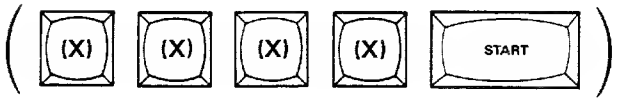
Fuses may be edited one at a time from the front panel, or in line editor fashion from a terminal. In the examples that follow, assume that we are editing the Basic Gates fuse map of figure 3-9, representing the logic diagram of figure 3-10.

If "Device Selection Error" appears when you enter the fuse editor, you must specify the family code and pinout code to define the fuse map.

Front Panel Control

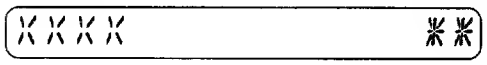
Enter the fuse editor with select code EE:




29A Displays

XXXX is decimal number of fuse being edited, ** is binary state of fuse number XXXX (00 or 01).



The desired fuse number for editing from the front panel may be scrolled to by using the START and REVIEW keys, or specified directly by entering the fuse number XXXX as shown above. The data display on the right reflects the current state of the selected fuse:

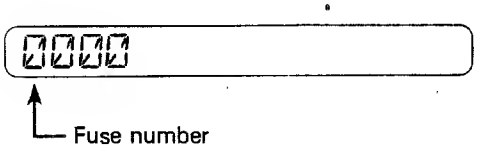
01 = high-resistance, "blown" fuse

00 = low-resistance, fuse intact

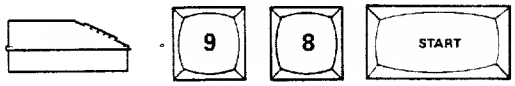
Entering a 0 or a 1 while displaying a selected fuse will store that state for the fuse.

To change fuse number 98 in our Basic Gates example from unblown to blown:

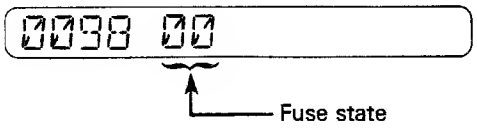

29A Displays



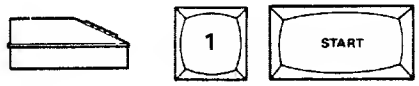

Enter the decimal fuse number, 98.



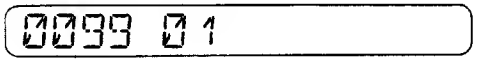

29A Displays




This display indicates that RAM data for fuse 98 is set for "don't program." To change it to a programmed (blown) state:



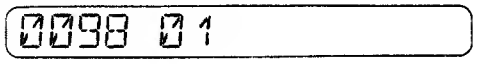

29A Displays




(Fuse number increments automatically.)

To decrement a fuse number:

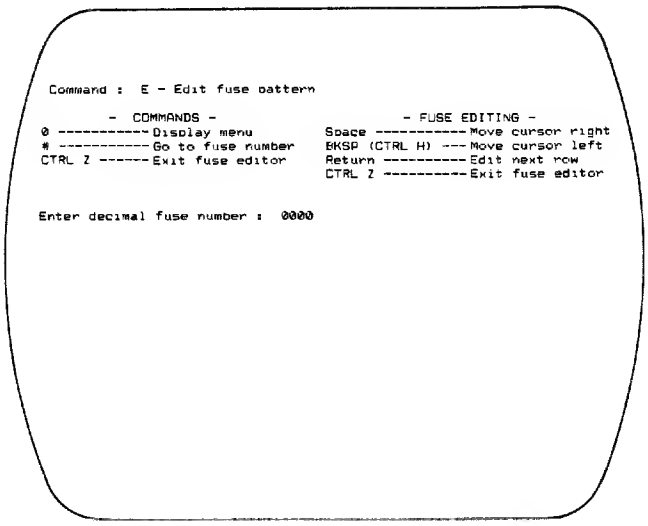

29A Displays




Terminal Control

Enter an "E" from the terminal Command mode:


Terminal Displays




You may now specify a fuse number directly, or enter RETURN to display the first fuse row.

The fuse editor is a fixed-format line editor, with the first column of the displayed line reserved for command characters. A character entered in the first column (normally blank) is interpreted as a command and acted upon immediately; otherwise fuse editing is not processed until a RETURN or CTRL Z is entered (at any point on the line). The command characters recognized in the first column are Ø (zero) and #.

The fuse editor display (see figure 3-15) shows the specified fuse number followed by the next N consecutive fuses, where N is the number of fuses in one row of the selected PAL. Any fuse number may be specified, regardless of row boundaries, and the display will follow this convention. Thus entering RETURN at any time moves the editor to the fuse one row down from the previously specified fuse. Index marks are shown over every tenth fuse in the row displayed, for easy location of fuses beyond the one specified. Also note that the fuse display may be changed from X/- to 0/1 with select code CE or main menu command G (see section 3.5.12).

In operation, the fuse editor copies the selected row to a temporary buffer where all editing changes are made. Then, when a command or RETURN is entered, the editing buffer is examined for legal characters before copying back to the fuse map. You are not allowed to proceed to another row until all characters are legal in the current row. Typing a CTRL Z to exit the fuse editor from an untested edited row will leave the row in its original state.

To edit a fuse row, use the following procedure:

1. Move the cursor back and forth along the displayed row using SPACE and BACKSPACE until it is positioned over the fuse to be changed.
2. Type the desired symbol to enter it into the editing buffer as the fuse state.
3. Enter 0 (zero) or # in the command character position at any time to display the menu or move to a specific fuse number.
4. Type RETURN at any time to move to the next row.
5. Type CTRL Z at any time to exit the fuse editor.

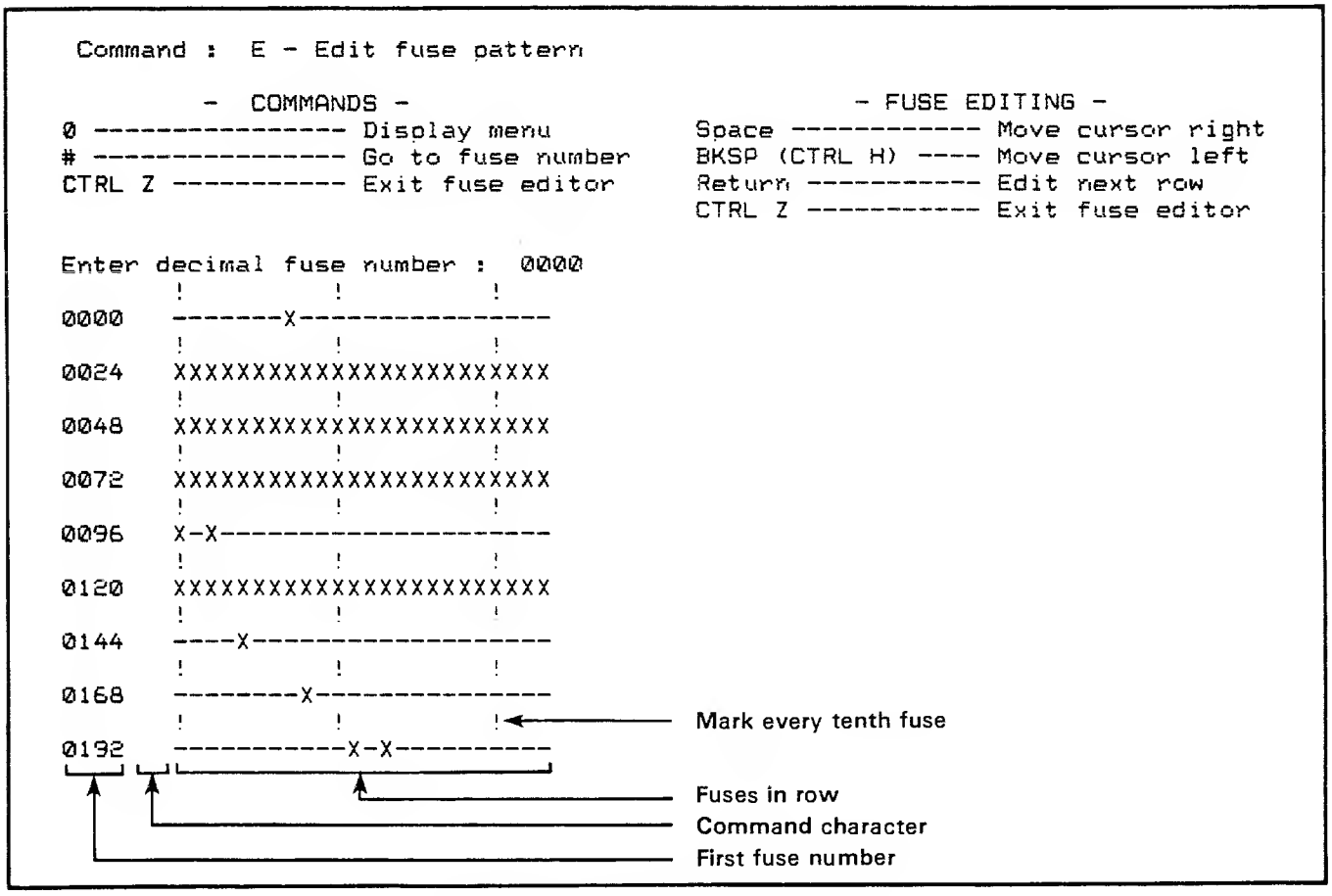



Figure 3-15. Default Fuse Editor Pattern

Editing fuse number 98 in our example may be done in two ways. As one method, you can enter the fuse editor and type RETURN until the desired row appears (beginning with fuse 0096), resulting in a display that matches the device data sheet, and then space three times to locate fuse 98. The display in this case will resemble figure 3-15.

Alternatively, fuse number 98 may be directly specified. When this is done, a fuse "row" is displayed which begins with fuse number 0098 and does not match any of the rows in the logic diagram of figure 3-10. Fuse number 98 may now be modified without counting spaces, and subsequent RETURNs will jump to the fuses directly below fuse 98 in the same column (122, 146, 170, etc.). Figure 3-16 shows the display when this method is used.

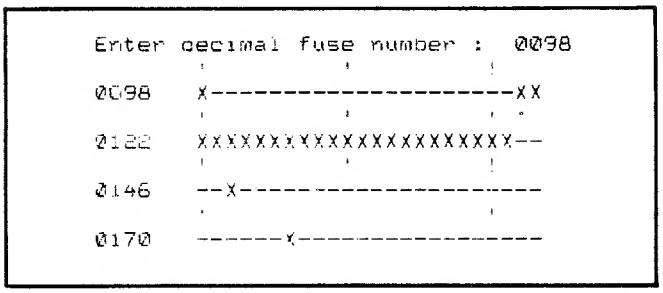



**Figure 3-16. Starting Fuse Not On Row Boundary**

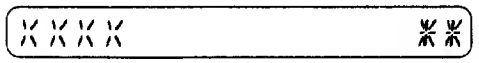
### 3.5.11 DISPLAY CONFIGURATION NUMBER

This command displays the configuration number of the adapter firmware. Configuration numbers are used as serial numbers for firmware.

#### Front Panel Control

Enter SELECT EF from the front panel:

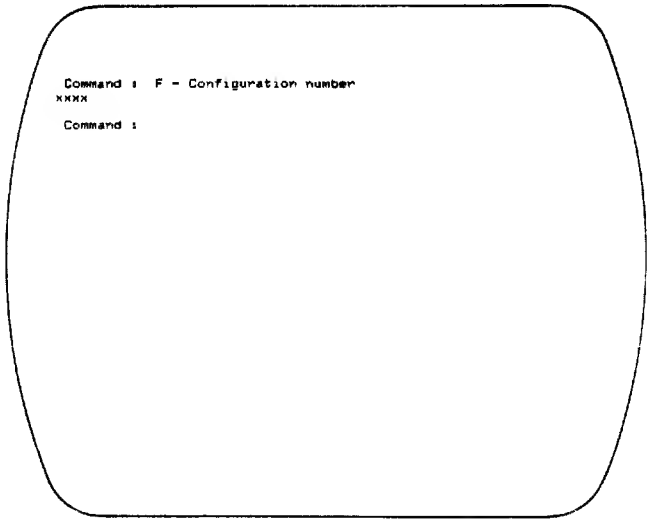

29A Displays



#### Terminal Control

Enter an "F" from the terminal command mode:


Terminal Displays




*NOTE*

*XXXX is the configuration number of the firmware in the adapter plugged into the PLDS.*

### 3.5.12 SELECT ATTRIBUTES

This command allows you to select one of two options for any of seven attributes as shown below. The only options available to the PALASM and H & L Design Adapters are those numbered Ø thru 7:

| OPTION | DESCRIPTION |
|---|---|
| Ø | Echo (full duplex): PLDS echoes all characters received at the serial port. |
| 1 | No echo (half duplex). |

*NOTE*

*The default echo mode will depend upon the programmer being used. The 29A and 100A programmers will power up in the "no echo" mode, while the Model 19 will power up in the echo mode.*

| | |
|---|---|
| 2 | JEDEC full mode: described by the JEDEC standard (JC-42-1-81-62). This is the default state. |
| 3 | JEDEC kernel mode: selects the kernel mode (see appendix A for kernel mode definition). |
| 4 | Fuse display X/—: displays an unprogrammed fuse as "X" and a programmed fuse as a "-". This is the default state. |
| 5 | Fuse display 0/1: displays an unprogrammed fuse as "0" and a programmed fuse as "1". |
| 6 | End upload with ETX: PLDS terminates an upload operation (serial data transmission) with an ETX character (ASCII hex Ø3). This is the default state. |
| 7 | End upload with CTRL Z: ends the upload with a CTRL Z. |

An underblow condition occurs when the programmer RAM indicates that a particular fuse should be blown and the device in the socket shows the fuse to be unblown. An overblow condition occurs when the programmer RAM indicates that a fuse is unblown yet the part shows it to be blown.

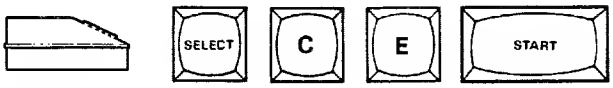
| | |
|---|---|
| 8 | Disable underblow/overblow display: disables this attribute. |
| 9 | Enable underblow/overblow display: enables this attribute. |

Some registered devices need to be initialized to a known state before functional testing. One method is to force the registers to a particular state. This function is referred to as preload. The default state (A) disables the function. The function may be enabled by entering a "B", but if it is not implemented in the P/T adapter, a warning message will be output and the feature once again is disabled.

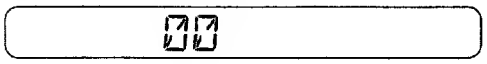
| | |
|---|---|
| A | Disable preload option: this is the default state. |
| B | Enable preload option |
| C | Two-pass functional verify: performs the normal two-pass functional verify at $V_{CC}$ voltages above and below nominal. |
| D | One-pass functional verify: speeds up the testing cycle by only doing a one-pass functional verify at the nominal $V_{CC}$ voltage. |

### Front Panel Control

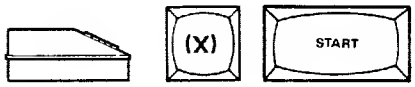
To access the attributes from the front panel, do the following:



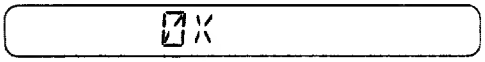

29A Displays



To change any attribute, enter the code number from those given above where the (X) is shown in the following key sequence.



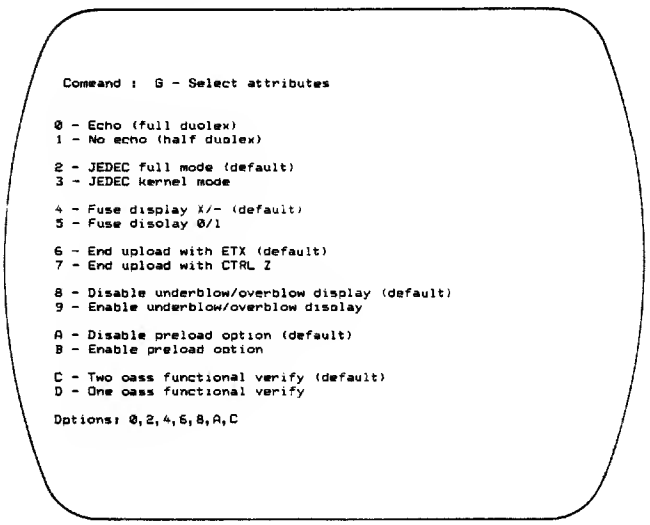

29A Displays



### Terminal Control

To access the attributes from the terminal, enter a G from the command mode.

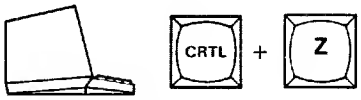

Terminal Displays




To change an attribute or attributes from the terminal, space or backspace (CTRL H) to the appropriate attribute(s) and enter the new value. The edit session is terminated by a RETURN if the edited attribute(s) are to be saved or by a CTRL Z if they are not to be saved. If an invalid value is entered the line will be repeated, including the invalid data, waiting for the correct value(s) to be entered.

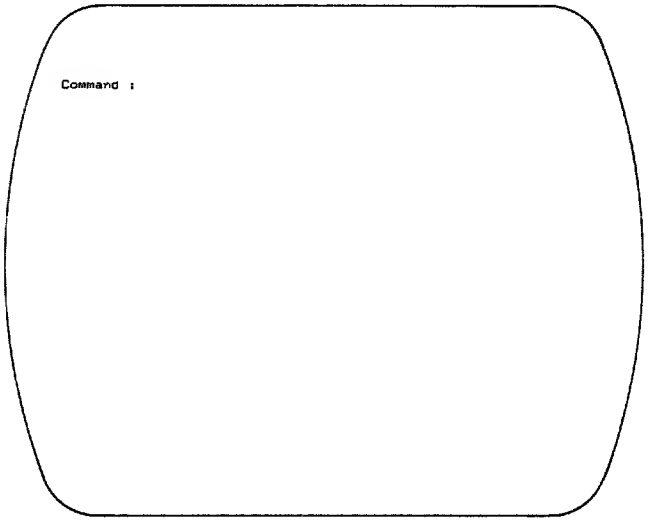
### 3.5.13 EXIT COMMANDS

During terminal mode, a CTRL Z will exit specific operating modes. When using the design adapters, this function is also used to terminate the change, insert, and edit modes.




simultaneously

Terminal Displays




The ESC key is used to terminate all PLDS operations and return control to the front panel. This must be done before removing an adapter or the LogicPak™.


returns control to the programmer front panel from terminal control.

# SECTION 4
# MAINTENANCE/CALIBRATION/ TROUBLESHOOTING

## 4.1 OVERVIEW

The support material in this section has been provided to help you keep your LogicPak™ in optimum operating condition. General maintenance practices are discussed in section 4.2, and the basic troubleshooting setup is described in section 4.4. For those LogicPak™ users who prefer to do their own calibration, detailed procedures, including measurement charts and timing diagrams for each device, are provided in each adapter manual. The basic procedures to set up the LogicPak™ in the calibration mode are described in section 4.3 of this manual.

## 4.2 MAINTENANCE

Regular maintenance of the LogicPak™ consists of cleaning (section 4.2.1) and inspection (section 4.2.2).

### 4.2.1 CLEANING

Inspect the LogicPak™ inside and out for accumulated dirt or dust. To clean the LogicPak™:

1. Wipe any dust or dirt off the outside of the LogicPak™ with a clean, damp cloth.

*NOTE*

*Do not use abrasive cleaners or solvents that will etch the paint.*

2. Remove dust from the circuit boards with a blast of dry, compressed air or a clean, soft-bristled brush.

### 4.2.2 INSPECTION

You can help prevent malfunctions by periodically inspecting your LogicPak™. Check cable connections, card seating, component mounting, etc., for shorts, opens, or discontinuities.

If you find heat-damaged components, be particularly careful to find and correct the cause of the overheating to prevent further damage.

## 4.3 CALIBRATION

The need for calibration varies with the amount of use your LogicPak™ receives. Generally, we suggest calibration whenever: 1) programming yields fall below the manufacturer's recommended minimums, 2) when troubleshooting has been completed, or 3) if the user's company policy requires periodic calibration certification.

Because the LogicPak™ must be calibrated with an adapter installed and the values vary with different adapters, the detailed calibration procedures, measurement charts and timing diagrams are provided in each adapter manual. The calibration setup procedure is described in this section.

*NOTE*

*If you do not have the equipment necessary to perform calibration and repairing procedures, contact the nearest Data I/O Service Center. Because of the different programmer mainframes and adapter modules, this manual does not attempt to cover all areas of programmer calibration. Instead, it lists the steps necessary to calibrate only the LogicPak™.*

The first step in testing the LogicPak™ hardware is to measure the programmer power supplies (and adjust if necessary). Proper LogicPak™ operation requires calibrated programmer power supplies. These measurements are essential before any tests are performed on the LogicPak™ itself. Refer to your programmer manual for power supply adjustment.

To prepare the LogicPak™ for calibration:

1. Remove the adapter (if any) from the LogicPak™.
2. Remove the four phillips-head screws from the LogicPak™ cover (see figure 4-1).
3. Remove the two allen screws on each side of the LogicPak™ cover (see figure 4-1).

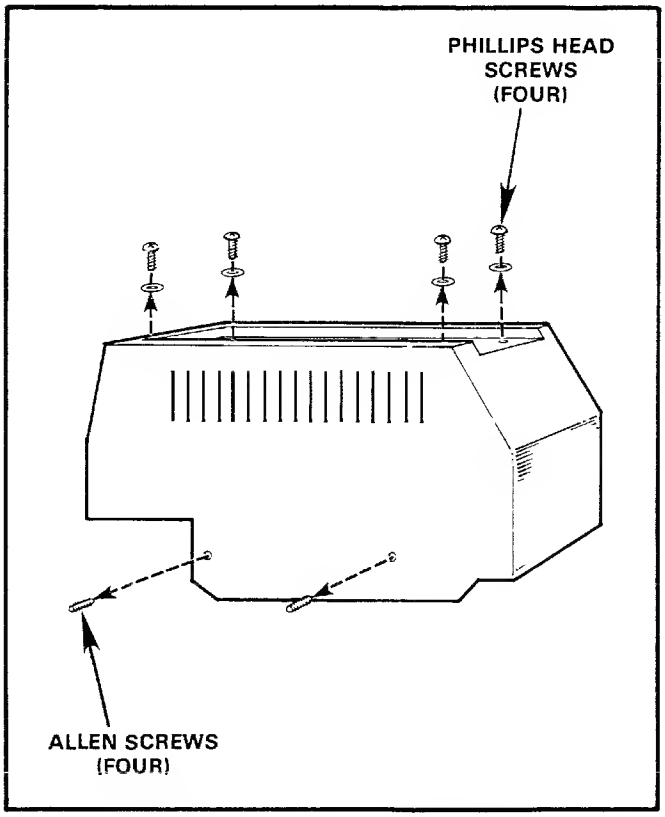



**Figure 4-1. LogicPak™ Cover Removal**

4. Lift the cover off the circuit board cage assembly.
5. Plug the adapter into the connector on the pin driver board as shown in figure 4-2.

To calibrate the LogicPak™ follow the series of steps presented in the measurement chart located in each adapter manual. The second step on the measurement chart is a "go/no-go" functional test of the module.

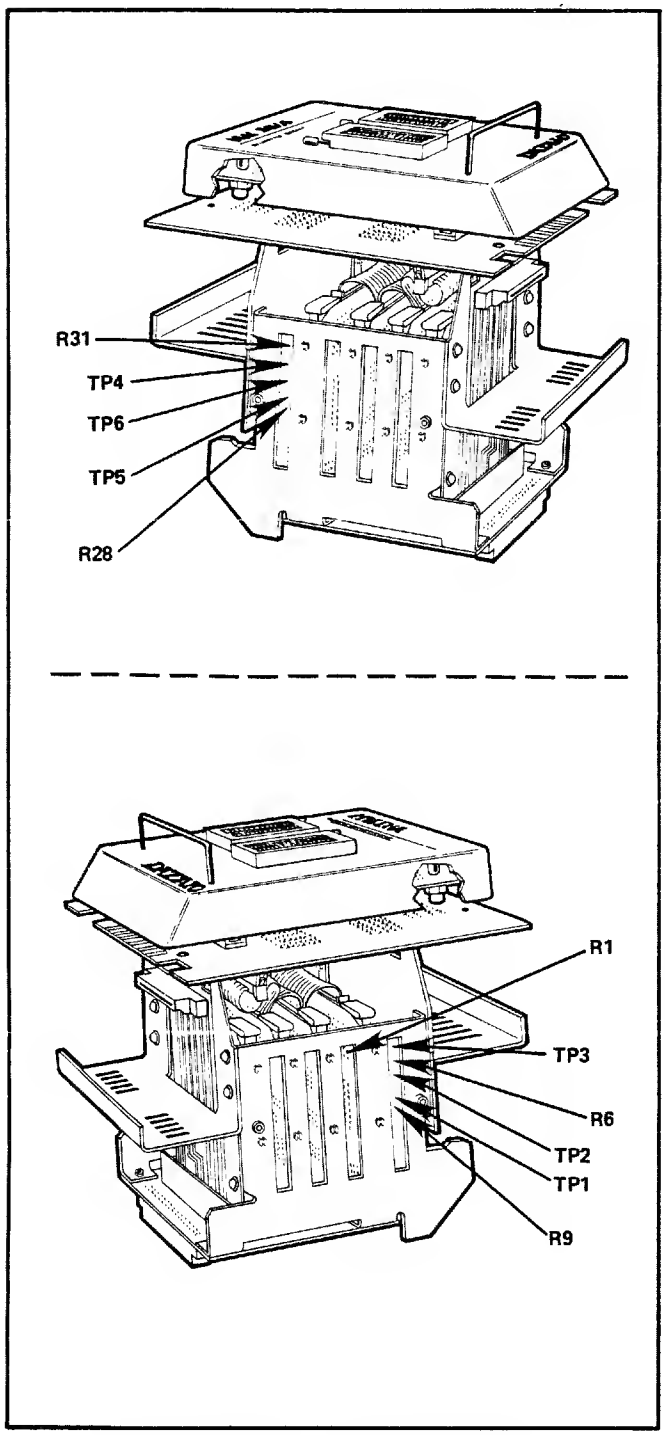



Figure 4-2. Calibration Equipment Setup

**CAUTION**

**If the LogicPak™ fails the second step on the measurement chart in the adapter manual, DO NOT proceed to the next step. The hardware must pass this step or further testing may damage the LogicPak™.**

Steps 1 through 4 and step 16 serve to calibrate the module. Potentiometers adjust each digital-to-analog converter (DAC) to the optimum voltage. Measurements are made at test points on the LogicPak™, shown in figure 4-3, or at socket pins on the P/T adapter.

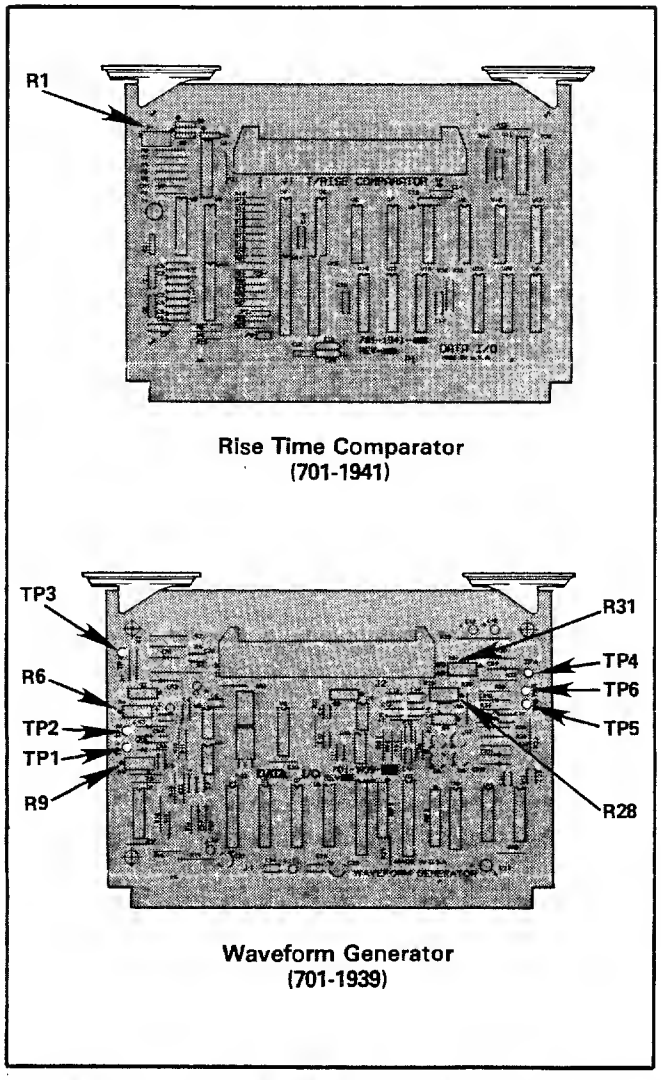



Figure 4-3. LogicPak™ Test and Adjustment Locations

If the LogicPak™ fails any step on the measurement chart in the adapter manual, do not continue to the next step. Refer to table 4-1, which lists suspect boards and components that would cause the module to fail for each step. Subsequent tests will not give valid results unless all preceding steps are passed and adjustments made.

Table 4-1. PLDS Error Codes

| ERROR CODE | DESCRIPTION | ACTION |
|---|---|---|
| 21* | Illegal Bit Error | The programmer is asked to leave intact a fuse which is already blown. Examine programmer RAM and the device's data. |
| 22* | Programming Error | An attempt to blow a fuse was made and failed. Try the programming sequence again. If the second attempt also fails, try another device. If both devices produce this error, check the calibration of the LogicPak™. If calibrated correctly contact your local Data I/O office. |
| 25* | No Socket Adapter | Insert appropriate socket adapter. |
| 30 | No (or Invalid) Device Selected | Enter valid device family and pinout codes (refer to Comparison Chart of Programmable Logic Device in each adapter manual). |
| 31* | Overcurrent | Hardware error in LogicPak™ or shorted device. Substitute a known-good device or consult the troubleshooting section. |
| 32* | Backward Device/$V_{CC}$ Overcurrent | (1) Device plugged in backward; turn it around.<br>(2) $V_{CC}$ overcurrent, probably caused by a faulty device. |
| 34 | Invalid Device Selected | Incorrect family pinout code entered. Enter proper family pinout code. This error occurs in computer remote control only. |
| 35 | Source Equation Translation Error | Check equation buffer by connecting terminal to examine the equation buffer. This error code lets the operator know that an error exists in the source equations when the programmer is not controlled by a terminal. |
| 36 | Begin RAM Pointer Not = 0000 | Refer to programmer manual to reset the begin RAM pointer to zero. This error usually occurs when changing from one programming pak to another. |
| 37 | Invalid Device-Related Operation | Verify, program, or other illegal operation was attempted, with a design adapter installed. |
| 38* | Calibration Step Error | (1) Indicates you have selected an incorrect calibration step, or<br>(2) a program operation is attempted prior to exiting calibration—exit the calibration mode (refer to the programmer manual). |
| 63 | RAM Write Error | System RAM failure. Refer to programmer manual or contact Data I/O service representative. |
| 65 | Firmware Sum-Check Error | Contact Data I/O service representative. This indicates that the EPROM firmware in the LogicPak™ or adapter may have changed since the unit was shipped. Do not continue operation until the situation is corrected. |

*These errors do not apply to design adapters.

Table 4-1. PLDS Error Codes (Cont.)

| ERROR CODE | DESCRIPTION | ACTION |
|---|---|---|
| 70* | DAC Error, $V_{CC}$ | See section 4.4 (troubleshooting). |
| 71* | DAC Error, Bit Switch Number 1 | See section 4.4. |
| 72* | DAC Error, Bit Switch Number 2 | See section 4.4. |
| 73* | DAC Error, CE | See section 4.4. |
| 74* | Logic Fingerprint™ Test Verify Error | Indicates one of the following Logic Fingerprint™ errors:<br>(1) Device passed fuse verify but failed Logic Fingerprint™ Test—defective device.<br>(2) Operator has entered wrong test-sum.<br>(3) Device cannot be tested with Logic Fingerprint™ (refer to P/T adapter manual for the limitations of the Logic Fingerprint™ test). |
| 75 | Structured Test Verify Error | The device passed fuse verify but failed structured test—defective device. Check structured test vectors and make sure they are correct. If not, reenter the correct vectors. The vector could be invalid, or the operator may have miskeyed a valid vector. |
| 76 | Self-Test Error | Indicates failure in the LogicPak™. Consult section 4.4 (troubleshooting) or contact your Data I/O service representative. |
| 77 | Security Fuse Programming Error | (1) Indicates that the security fuse option cannot be programmed in the installed device, or (2) there is no security fuse option available for this type of device. |
| 78* | No Fuse Verify Set | Indicates you have tried to program the device with the verify-option mode set for 2. The verify option will not allow this. When this error code displays, select E6 and enter Ø or 1, and then you will be allowed 1 program. |
| 79* | Preload Not Implemented | The preload algorithm is not implemented for this device. |
| 81 | Parity Error | A parity error occurred while receiving serial data. |
| 82 | Checksum Error | Indicates an incorrect transmission data from a peripheral to the serial port, including fuse data, CRs, STX, etc. |
| 84 | Invalid Data | See section 3.5.9. |
| 91 | Fuse Address Error | See section 3.5.9. |

*These errors do not apply to design adapters.

## 4.4 TROUBLESHOOTING

This section will help you interpret and isolate failures in the LogicPak™. Use it in conjunction with the calibration section in each adapter manual, section 5 (Circuit Description) and the schematics provided in the back of this manual.

There are three major classes of failures that can occur in a system composed of a programmer and a LogicPak™. The first is no system operation, the second is poor yields, and the third is LogicPak™ failure.

After successfully troubleshooting the LogicPak™, it must be calibrated according to the instructions in each adapter manual. It is very important that you calibrate the programmer before you calibrate the LogicPak™. Refer to your programmer manual for detailed programmer calibration procedures.

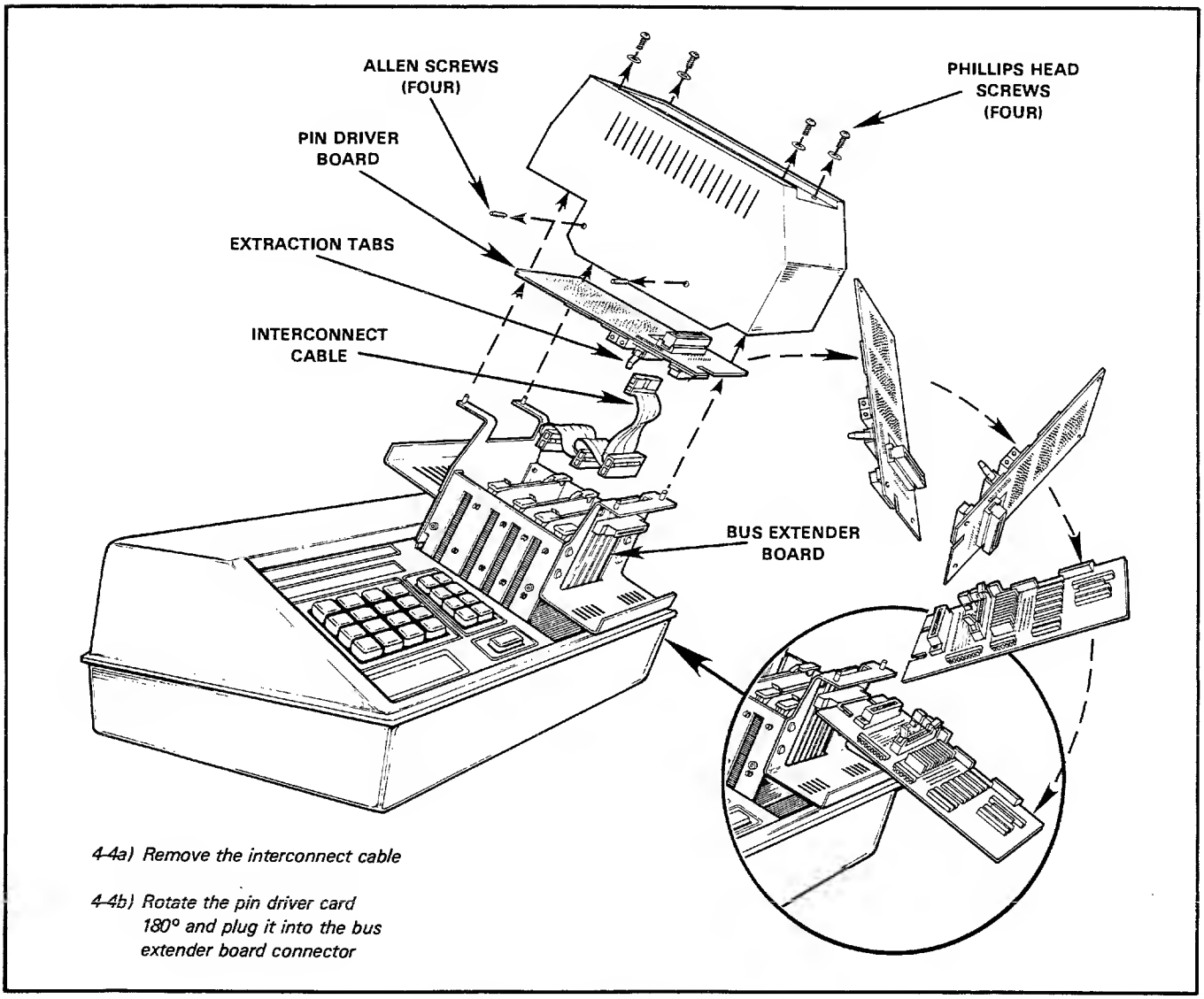
### 4.4.1 EQUIPMENT SET UP

The following equipment is necessary to troubleshoot the LogicPak™:

- Data I/O extender card (part number 701-1949) and service cable (part number 709-0078)
- one phillips screwdriver
- one 1/16-inch allen wrench

To prepare your LogicPak™ for troubleshooting:

1. Turn off the programmer power (section 3.3).
2. Remove the LogicPak™ from the programmer (section 2.3).
3. Remove the adapter (if any) from the LogicPak™.
4. Remove the four phillips-head screws from the LogicPak™ cover (see figure 4-4).




*4-4a) Remove the interconnect cable*

*4-4b) Rotate the pin driver card 180° and plug it into the bus extender board connector*

**Figure 4-4. LogicPak™ Disassembly**

5. Remove the two allen screws on each side of the LogicPak™ cover (see figure 4-4).
6. Lift the cover off the card cage assembly.
7. Unplug the pin driver board (702-1943-001) from the bus extender board and lift it slightly until you can see the interconnect cable and its connector (figure 4-4).
8. Flip the extraction tabs out on each side of the connector (figure 4-4).
9. Pull the cable out of the connector.
10. Rotate the pin driver board 180° and plug it into the connector on the front of the bus extender board (figure 4-5).

*NOTE*

*The interconnect cable must be replaced by the service cable before you can troubleshoot your LogicPak™.*

11. Lift the extraction tabs on each side of the three forward circuit cards as shown in figure 4-4.
12. Unplug the interconnect cable from the three connectors shown in figure 4-4.
13. Lift the extraction tabs on the waveform generator card (figure 4-5).
14. Pull out the waveform generator board along the guides (figure 4-5).
15. Gently push the service extender card (701-1949) along the guides to plug it into the J2 connector on the motherboard (702-1938) where the waveform generator was located.

*NOTE*

*Any board can now be placed on the extender board. The waveform generator card (701-1939) must be either on the extender board or in the slot just behind it.*

16. Firmly, but gently, push the service cable into the connectors on the three boards (as shown in figure 4-5). Notice that the extraction tabs will move back to their locked positions when the cable is locked into the connector.
17. Gently push the extender cable into the connector on the pin driver board (see figure 4-5).

### 4.4.2 NO SYSTEM OPERATION

The following steps should be performed if the system will not initialize with the LogicPak™ installed. After completing each step, determine whether the problem still exists.

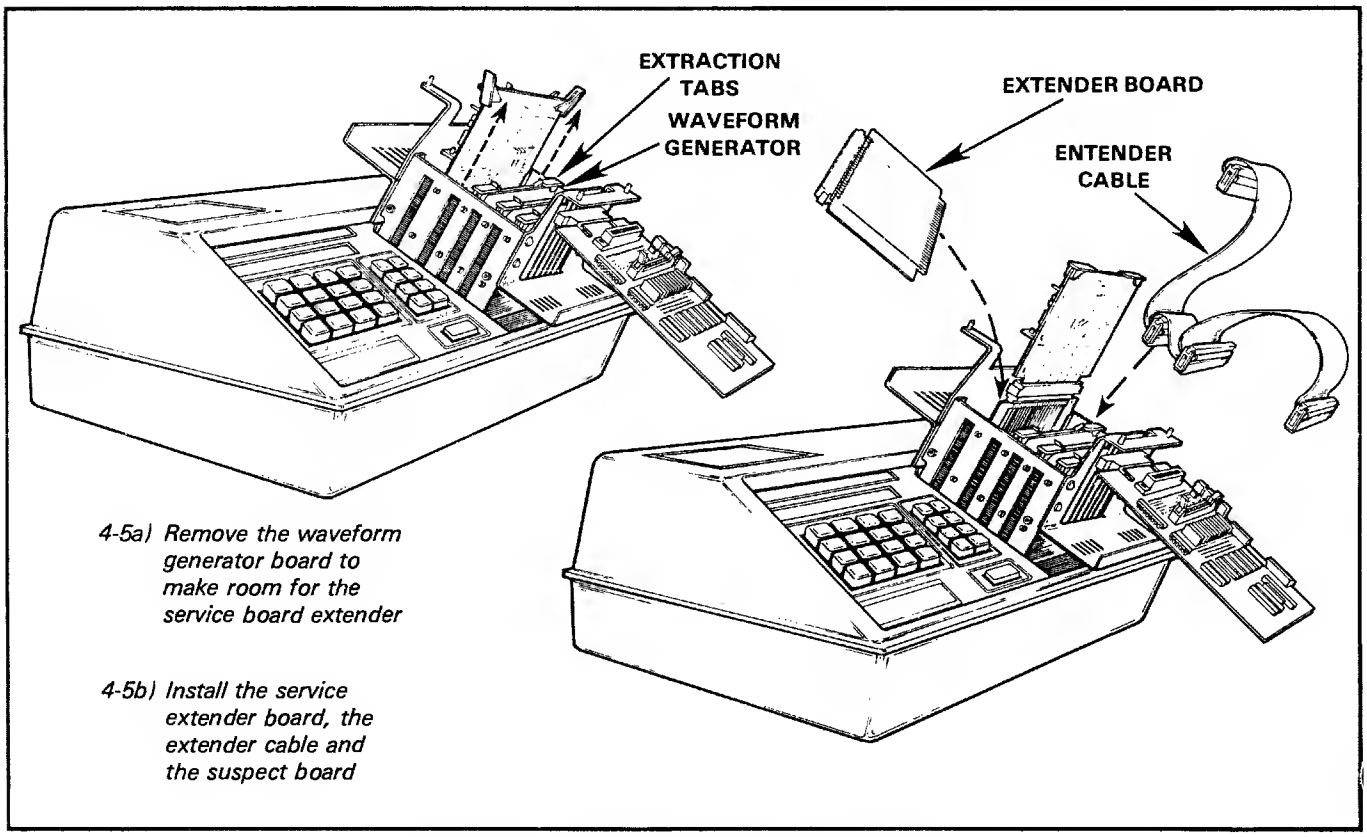



**Figure 4-5. Troubleshooting Equipment Setup**

1. Check to be sure the LogicPak™ and adapters are properly installed in your programmer (see section 2.4).
2. Check to be sure the power cord is properly connected (see section 3.2).
3. Check the programmer mating connector (J1) for bent or broken pins (Pin HH is intentionally shorter).
4. Check the cards to be sure they are correctly installed in their connectors (section 4.4.1).
5. Check the ribbon cable to be sure it is properly inserted in the connectors (section 4.4.1).
6. Check the programmer power supplies for proper voltage output levels (see programmer manual).
7. If steps 1 through 5 fail to isolate the problem, contact your local Data I/O Service Center.

### 4.4.3 LOGICPAK™ FAILURE

The following steps should be performed if a device will not program at all or if error messages are displayed. After completing each step, determine whether the problem still exists.

1. Check that the family and pinout codes are correct for the device and that the device is being inserted in the correct socket (the one with the illuminated LED).
2. If possible, try to program a known-good device to determine if it is a hardware or device problem.
3. Check to be sure the LogicPak™ and adapters are properly installed.
4. Check the programmer mating (J1) connector for bent or broken pins (Pin HH is intentionally shorter).
5. Check the LogicPak™ cards to be sure they are correctly installed in their connectors (section 4.4.1).
6. Check to be sure the ribbon cable is correctly and properly inserted in the connectors.
7. Perform a complete calibration noting any measurements falling outside of the indicated limits. Refer to the calibration section in each adapter manual, as well as the Circuit Description (section 5) and the schematics in this manual to attempt to isolate the problem.
8. Perform waveform observations and note any discrepancies. Referring to the Circuit Description and the schematics may be helpful in isolating the problem.
9. Refer to the section on limitations of the Logic Fingerprint™ test in the P/T adapter manual to see whether any of those conditions apply to the problem at hand. A Logic Fingerprint™ test that gives multiple test-sum for a single device is an indication of an oscillating condition.
10. If steps 1 through 9 fail to isolate the problem, contact your local Data I/O Service Center.

### 4.4.4 LOGICPAK™ ASSEMBLY

To remove the troubleshooting equipment and reassemble the LogicPak™, proceed as follows:

1. Lift up the extraction tabs on the cable connector on the pin driver board and the three boards in the card cage.
2. Remove the service cable from the LogicPak™.
3. Unplug the board from the connector on the service extender board.
4. Unplug the service extender board from the motherboard.
5. Using the flat surfaces of the extraction tabs, gently but firmly push the waveform generator board along the guides until it locks into the J2 connector on the motherboard; the extraction tabs will lock into position when the board is properly inserted.

*NOTE*

*Because of mechanical restrictions the waveform generator board can only be inserted into the J2 or J3 connector on the motherboard. The other three boards are interchangeable.*

6. Gently but firmly insert the normal interconnect cable into the connectors on the circuit boards.

*NOTE*

*The extraction tabs will move back to their locked positions when the cable is locked into the connector.*

7. Unplug the pin driver board from the bus extender board.
8. Rotate the pin driver board back 180° (the opposite of what is shown on figure 4-4) and insert the interconnect cable into the connector on the pin driver board.
9. Plug the pin driver board into the bus extender board as shown in figure 4-4.
10. Lower the LogicPak™ cover onto the assembly.
11. Replace the four allen screws and four phillips head screws to secure the cover in place.

# SECTION 5

# CIRCUIT DESCRIPTION

## 5.1 INTRODUCTION

This section defines the functions of the principal hardware components of the LogicPak™. Each major function is depicted by a block diagram accompanied by a written description.

## 5.2 GENERAL ARCHITECTURE

The principal components of the LogicPak™ are illustrated and labeled with their names and part numbrs in figure 5-1. Figure 5-2 is a block diagram of the LogicPak™ electronics. The LogicPak™ controlled by the programmer's extended processor bus (J6 in figure 5-2) through the programmer's mating connector.

LogicPak™ adapters interface with the LogicPak™. When these adapters are installed, the LogicPak™ provides all the specific hardware and firmware required to support numerous families of programmable logic. Support provided includes methods of specifying the logic device design, fuse programming and verification, and functional testing.

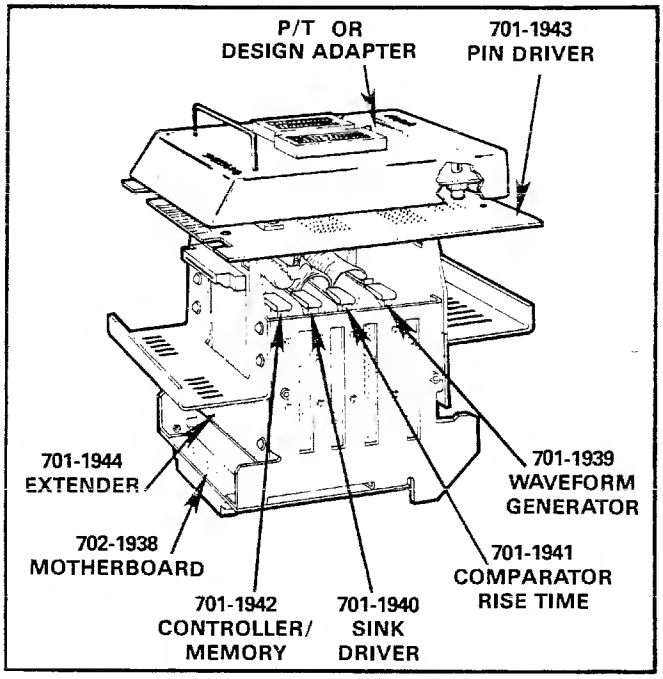



Figure 5-1. Principal Components of LogicPak™

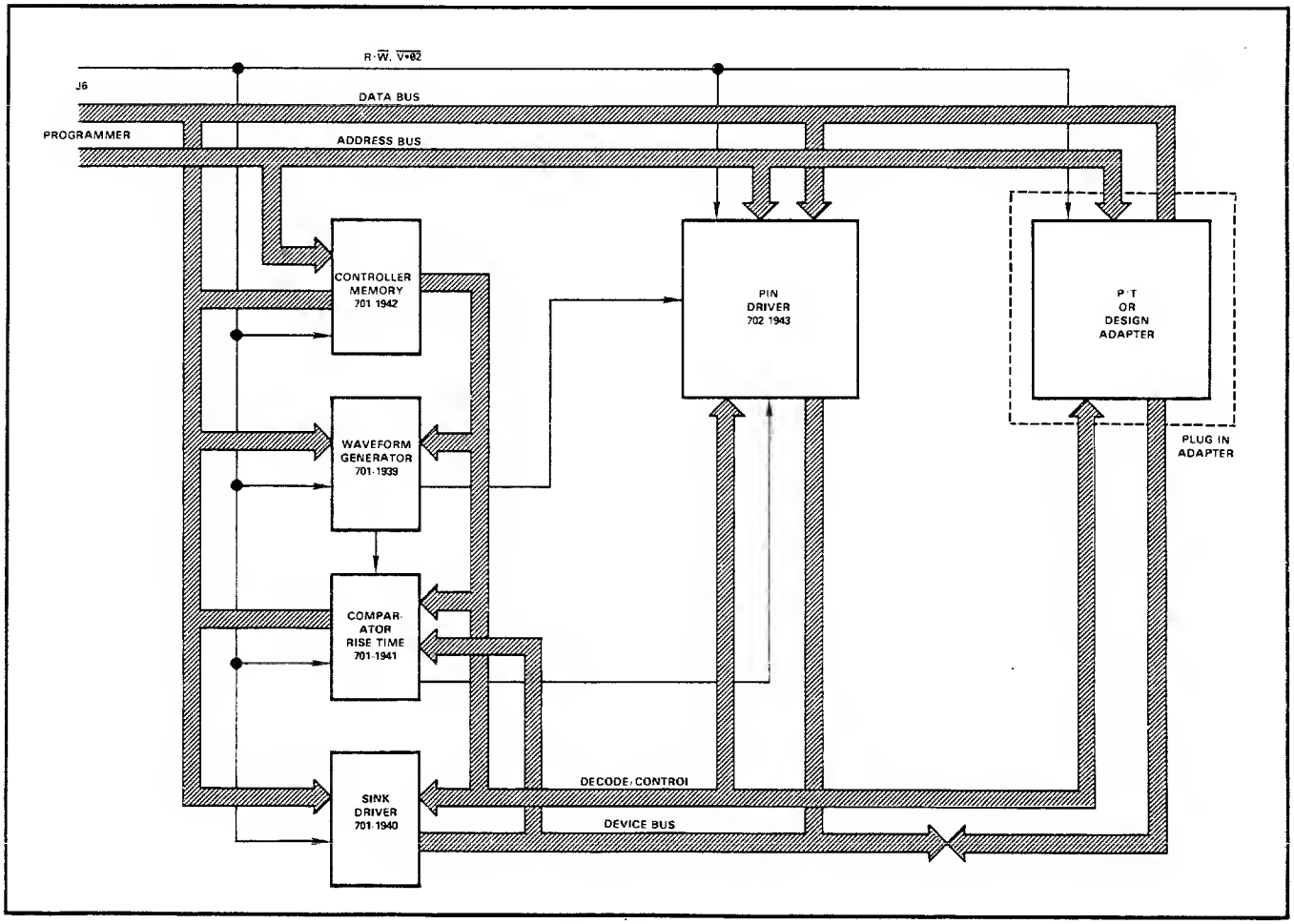



Figure 5-2. LogicPak™ Electronics Block Diagram

## 5.3 COMPONENT LAYOUT

Each board is described, and block diagrams are presented in sections 5.3.1 through 5.3.7. Schematics are provided at the back of this manual.

### 5.3.1 MOTHERBOARD

The motherboard (702-1938) accepts the programmer processor bus and power supplies and routes them to the five 72-pin connectors. It also serves as the interconnect between the five connectors for signals generated on the plug-in boards.

### 5.3.2 BUS EXTENDER

The bus extender board (701-1944) is installed in one of the five motherboard connectors and extends motherboard pins up to the top-mounted pin driver board (701-1943).

### 5.3.3 CONTROLLER/MEMORY

The controller/memory board (701-1942) has two functions: 1) it provides high-speed control operations that are not obtainable under processor operating speed, and 2) it contains the address decode circuitry that specifies the memory map for the base LogicPak™ and adapters. The block diagram of the controller/memory board is in figure 5-3.

**Control function.** A high-speed controller is necessary to supplement the programmer's processor because of the large quantity of serial data that must be read during programming, verifying, and testing of logic devices; see the controller portion of figure 5-3.

A free-running crystal-controlled oscillator produces a 4-MHz clock pulse that synchronizes the on-board operations of the control counter, vector counter, synchronize register, and control register. Its output is routed to other boards via the motherboard.

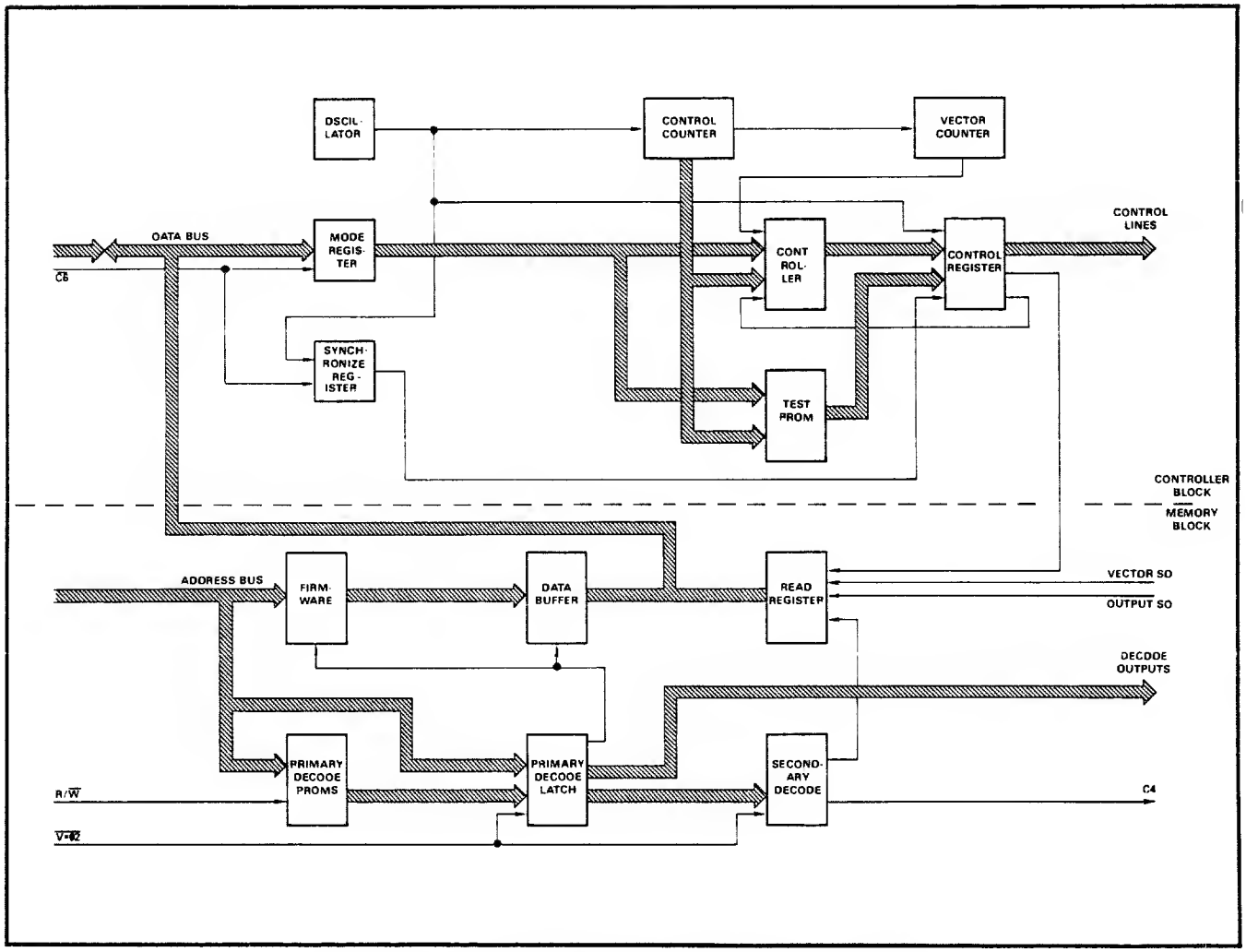



**Figure 5-3. Controller/Memory (701-1942) Block Diagram**

The mode register stores instructions from the processor that are presented to the controller and test PROM; these instructions describe which high-speed cycle the controller should perform. During each cycle, the control counter addresses the controller and the test PROM. Their decoded output function determines what control states should exist for that cycle. These states, which are in the form of registered control lines, are routed to other cards to control the input and output register operations. Each cycle is executed by output from the synchronize register. This output synchronizes the processor operations to the controller cycles. This synchronization is a handshake, busy/ready-type operation.

At the end of each cycle, data serially stored (under controller operation) in the read register are available for interrogation by the processor. These data represent: 1) logic device fuse states after fuse-related operations, 2) device output data during functional test operations, or 3) the Logic Fingerprint™ test.

During the Logic Fingerprint™ test, a vector counter tallies 128,000 cycles and then outputs a signal that tells the processor that one Logic Fingerprint™ test cycle has been completed. The test can then end or continue, based on user-selectable options. The test options range from Ø (no test performed) up to 99 cycles.

**Memory function.** The two parts of the memory function are: 1) primary decode, and 2) storage of LogicPak™ firmware; see the memory portion of figure 5-3.

1. **Primary decode.** The decode PROMs and decode latch allocate the available 16k bytes of processor memory to LogicPak™ functions. Where necessary, the output from the decode latch is routed to the secondary decode on other boards. Decode outputs are timed with V•Ø2 such that the selected locations are strobed to the data bus only during V•Ø2 time.
2. **LogicPak™ firmware.** The low-order 8k bytes of the 16k byte available space are allocated for LogicPak™ firmware. This firmware is stored in an 8k byte PROM whose outputs are buffered and presented to the data bus.

### 5.3.4 WAVEFORM GENERATOR

The waveform generator board (701-1939) selects specific voltage levels required to perform logic device fuse read, fuse program, and functional test operations. Figure 5-4 is a block diagram of the waveform generator board.

The three major power supplies of the waveform generator are: $V_{CC}$ supply, CE supply, and BIT supply. Each supply is under software control via a DAC (digital-to-analog converter). The DAC voltage output levels are obtained by presenting data via the data latch to the DAC inputs. This latch provides buffering and the timing required to store data. A common DAC reference voltage is generated from the DAC reference voltage circuit.

Each DAC output is routed to a power amp to provide for the high current requirements of the supplies. The power amps are op-amps with their positive voltage rails supplied through current sense detectors. The output rise and fall times are fixed by the slewing rate of the op-amp and are routed to the rise time section of the comparator/rise time board.

The current sense detectors have two selectable ranges. The ranges are controlled by the values stored in three bits of the control register. When activated, these detectors set the reset memory, which in turn kills all supply outputs by shutting down the DAC reference. The state of the reset memory is gated onto the data bus, and, if the memory state is set when inspected, an overcurrent error is flagged. The power-on reset circuit sets the reset memory when the programmer is powered up. When a device-related operation is selected, this memory is cleared by an initialize routine and remains cleared until an overcurrent condition sets the memory or the programmer is powered down. The current sense integrator smooths the transient overcurrent pulses, thereby avoiding false overcurrent errors.

Other functions of the control register are scope synchronization and provision of control lines for the adapter. These lines control the adapter LEDs, the backward device test circuitry, and the special clamping levels.

The comparator reference supply is identical in operation to the three other supplies except it does not require (or incorporate) a power amplifier and current sense stage. Its output is routed to the comparator section of the comparator/rise time board and is used as a programmable voltage reference level for the comparators.

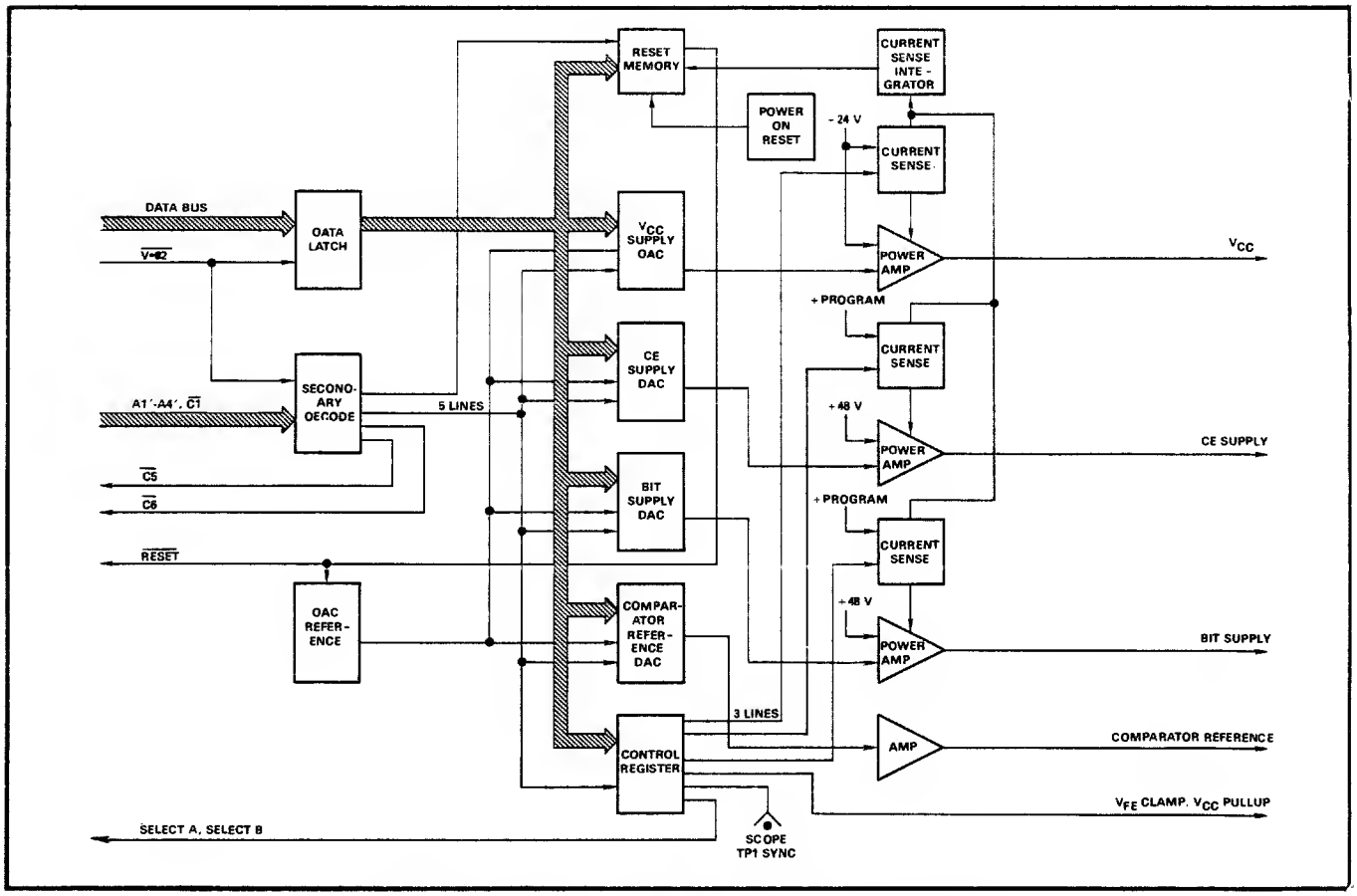



Figure 5-4. Waveform Generator (701-1939) Block Diagram

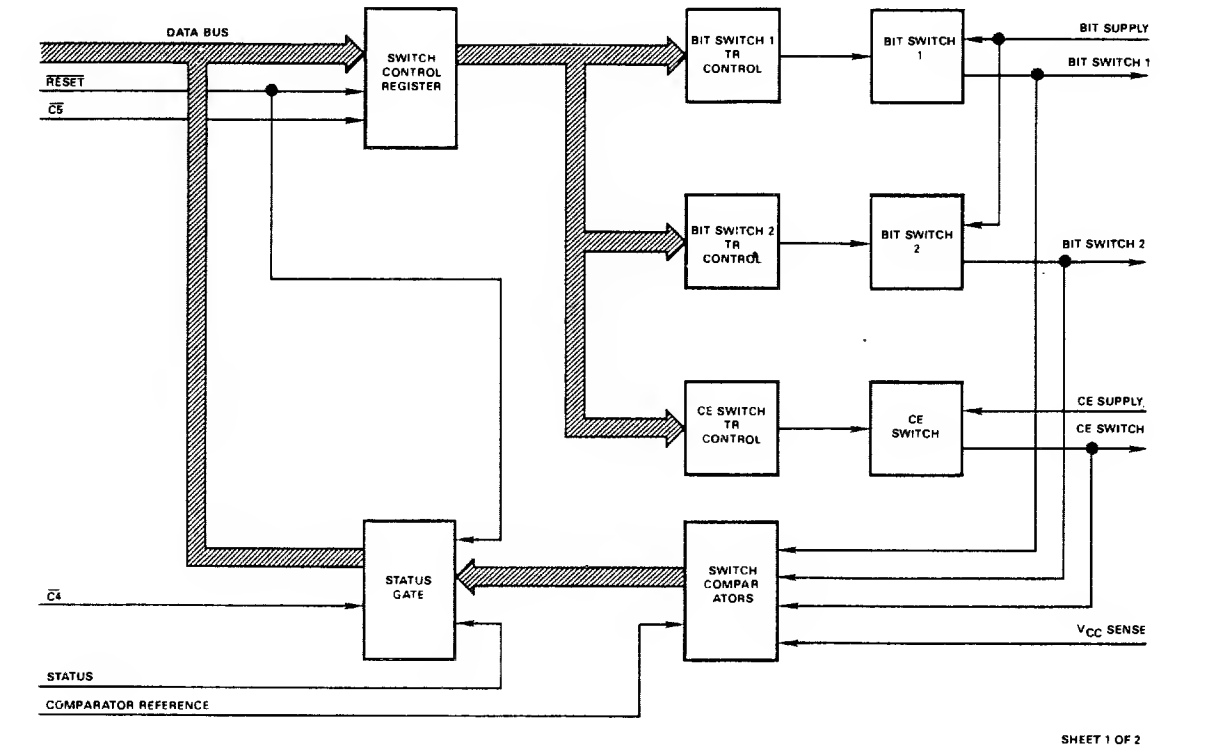
### 5.3.5 COMPARATOR/RISE TIME

The comparator/rise time board (701-1971) has two functions: 1) it monitors the logic device pins and the waveform generator outputs, and 2) it converts the waveform outputs into switchable signals with programmable rise time control. Figure 5-5 is a block diagram of the comparator/rise time board.

**Rise time function.** The switch control register determines which rise times are selected and which switches are to be activated. Each rise time control circuit consists of a current source (one common current value for all sources) driving the base of its switch transistor to charge the base capacitance. When the bases are released, a linear ramp is generated that is truncated at the switch supply level. The slew (dV/dt) rate is dependent on the value of capacitance switched to ground by the control register and the value of the current source.

The switch outputs and the nonswitched $V_{CC}$ sense line are connected to the switch comparator. This comparator allows the switch and $V_{CC}$ functions to be output as status to the status gate. Controller-ready status along with the switch/$V_{CC}$ status are monitored on the data bus. Error codes are displayed if the switches or $V_{CC}$ are not functional during the LogicPak™'s self-test.

**Comparator function.** The comparator monitors the pins of the logic device and compares them against the comparator voltage reference level. The comparators support the high-level program voltages and the TTL read levels of the device. Their outputs (device data) are stored in the output registers that, in turn, output data over the serial output (SO) line to the read register on the controller board. The register load and serial shift operations are synchronized to the controller operations by the control line (S/L) and oscillator (CP). As previously explained in section 5.3.3 (control function), this is how the logic device fuse and functional status are read.



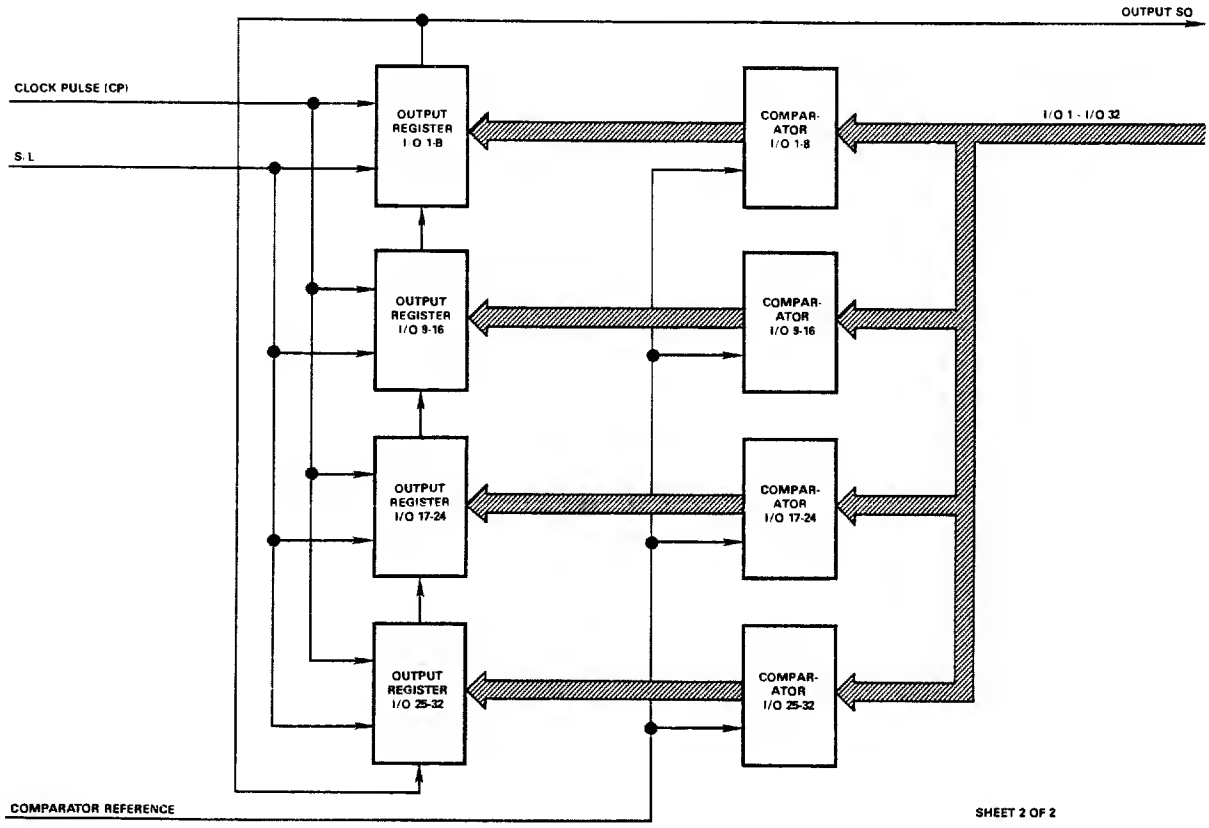




**Figure 5-5. Comparator/Rise Time (701-1941) Block Diagram**

### 5.3.6 PIN DRIVER

The pin driver board (702-1943) serves as a receptacle for the LogicPak™ socket adapters and contains the source driver circuitry that routes the switch outputs from the comparative rise time board (BIT switch 1, BIT switch 2, CE switch) to the socket adapter receptacle. Figure 5-6 is a block diagram of the pin drive board.

The source drivers are PNP transistor switches driven by current sources that are controlled by the source registers. The current sources provide a constant base drive to the PNP switches at all switch voltages.

The data latch provides the buffering and the timing required to store data in the CMOS source registers. This data storage will control the routing of the BIT switch and CE switch to the socket adapter receptacle.

The pin driver board also connects the sink driver board outputs and the extended processor bus extender board outputs to the socket adapter receptacle.

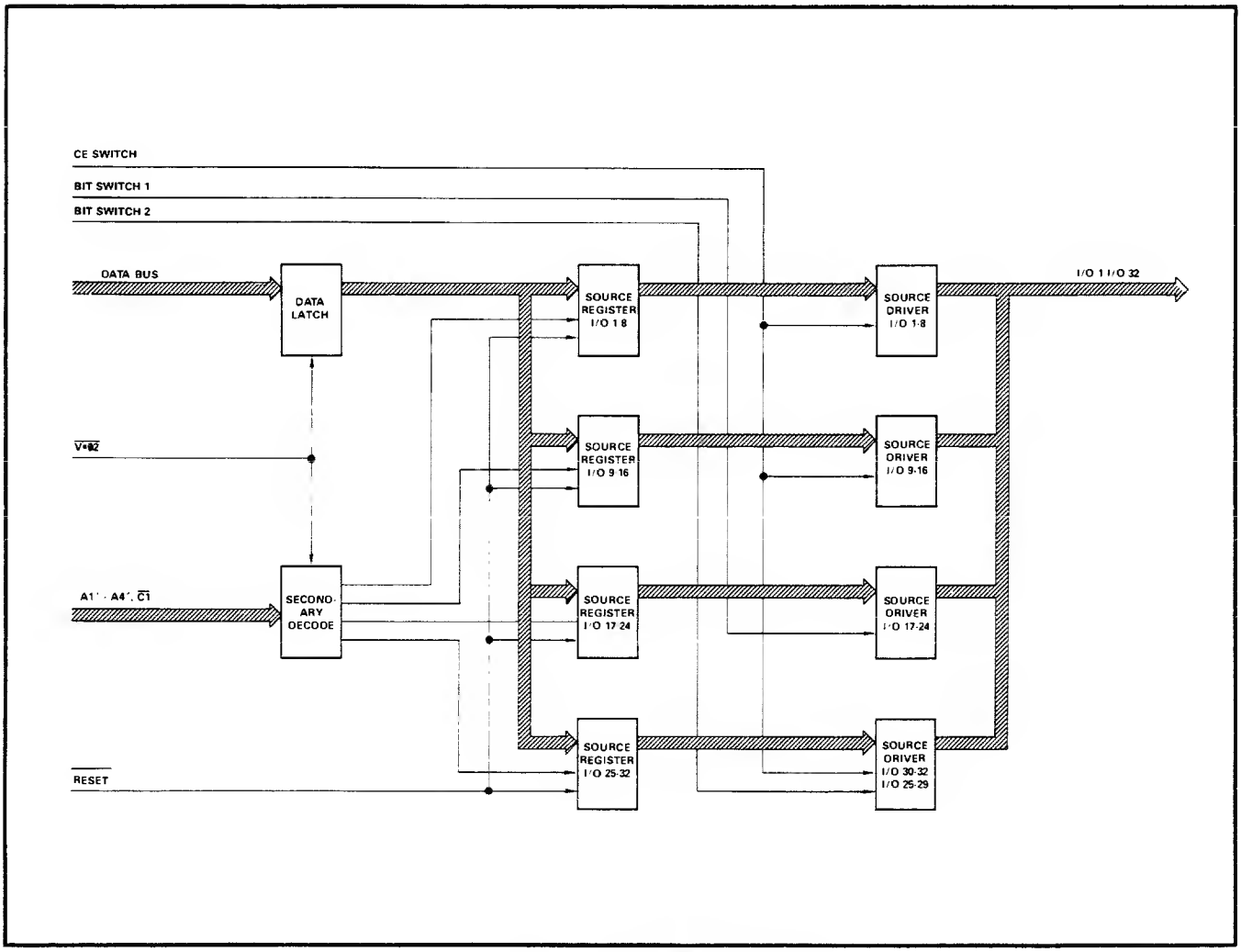



**Figure 5-6. Pin Driver (702-1943) Block Diagram**

### 5.3.7 SINK DRIVER

The sink driver board (figure 701-1940) provides the TTL levels that are necessary to read and program device fuses and perform function testing. The sink drivers are high-voltage noncommitted collector drivers. Their output drive is limited so that during a functional test, the logic device can override. Figure 5-7 is a block diagram of the sink driver board.

The input registers control the level of the sink drivers. These registers are under software control for fuse and structured test operations and are under controller board control for Logic Fingerprint™ test operations.

In Logic Fingerprint™ test operations, the registers are preloaded by using the starting vector specified. The Logic Fingerprint™ test cycle then begins, starting a serial shift operation of the registers with serial data input from the parity generator. Following 32 clock cycles (one controller cycle), the next Logic Fingerprint™ test vector is presented, and the register will halt until the logic device pins are tested. This procedure will continue until the controller board signals that the Logic Fingerprint™ test is then read by additional shift cycles with the serial data input being that of the register serial output (recirculate). The shift modes are controlled by control lines IRC and CP (see figure 5-7) and the data recirculation path is enabled by control line RICR.

The CP and CE multiplexer changes control of up to four sink drivers from input register control line to controller board control. This alternate control is necessary during the Logic Fingerprint™ test to provide special consideration to logic devices that have dedicated clock, reset, preset, and enable pins.

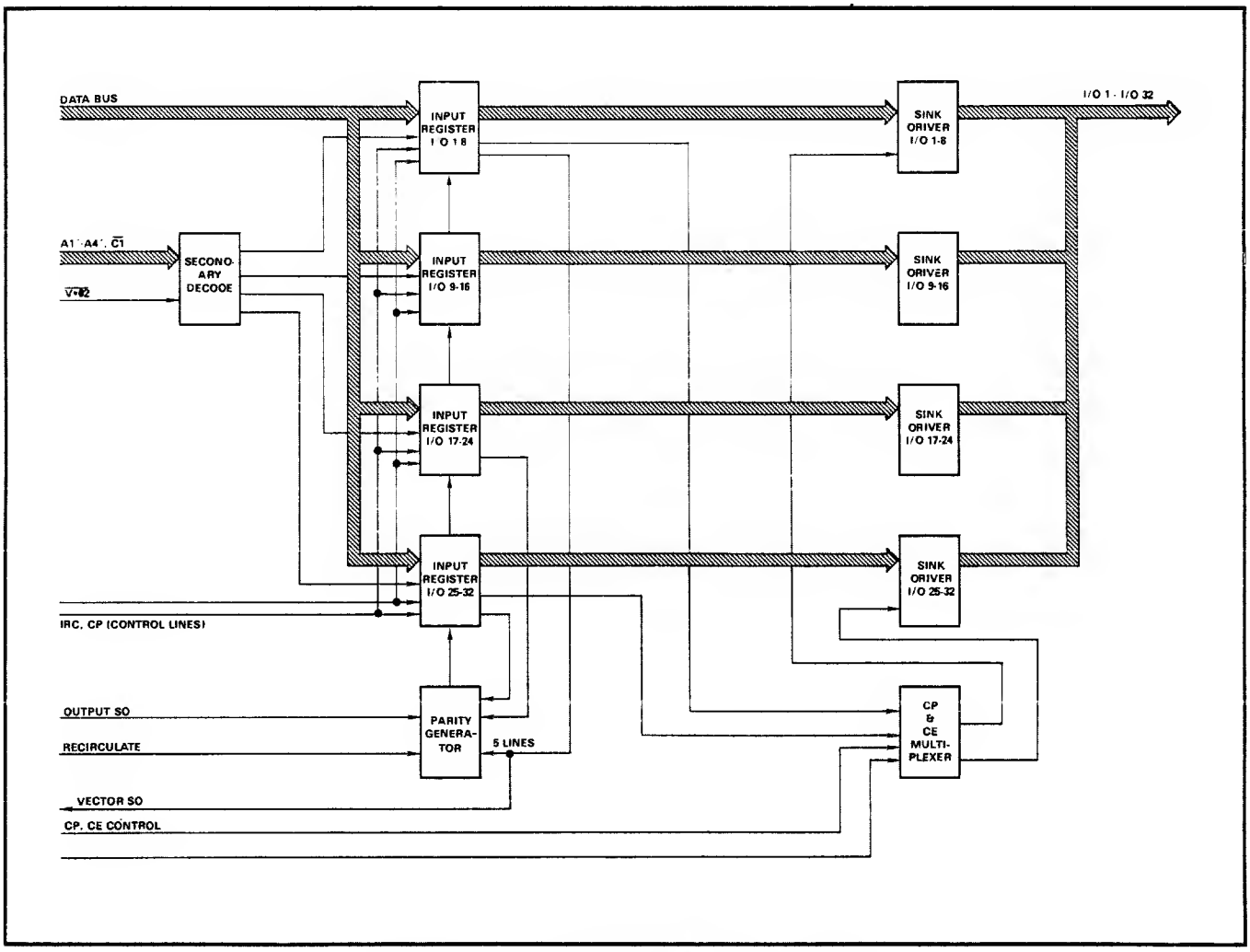



Figure 5-7. Sink Driver (701-1940) Block Diagram

# APPENDIX A

## STANDARD DATA TRANSFER FORMAT BETWEEN DATA PREPARATION SYSTEM AND PROGRAMMABLE LOGIC DEVICE PROGRAMMER

This document defines a format for the transfer of information between a data preparation or storage system and a device programmer. This format provides for, but is not limited to, the transfer of fuse, test, identification, and comment information in an ASCII representation. This format defines the "intermediate code" between device programmers and data preparation storage systems.

*NOTE*

*This is Data I/O's implementation of the JEDEC (Joint Electron Device Engineering Council) standard (JC-42.1-81-62).*



Data I/O Corporation
10525 Willows Road N.E./C-46
Redmond, WA 98052
(206) 881-6444

Version 1.1 May 5, 1983

# SECTION A.1

# INTRODUCTION

The Backus-Naur Form (BNF) is used in this document to define the format syntax.

## A.1.1 BNF RULES

"::=" denotes "is defined as".

Characters enclosed by single quotes are literals.

Parentheses enclose indentifiers.

Square brackets enclose optional items.

Braces (curly brackets) enclose a repeated item. The item may appear zero or more times.

A vertical bar separates alternative items.

A repeat count is given by a ":n" suffix. For example, a six digit number would be defined as (number) ::= (digit) :6.

Example

The English description of a person's name may be as follows:

> The full name consists of an optional title followed by a first name, a middle name, and a last name. The person may not have a middle name or may have several middle names. The titles consist of : Mr., Mrs., Ms., Miss, and Dr.

The BNF definition would be:

(full name) ::= [(title)] (f. name) { (m. name)}
| (l. name)

(title) ::= 'Mr.' | 'Mrs.' | 'Ms.' | 'Miss' | 'Dr.'

## A.1.2 BNF DEFINITIONS

(digit) ::= '0' | '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9'

(hex-digit) ::= (digit) | 'A' | 'B' | 'C' | 'D' | 'E' | 'F'

(binary-digit) ::= '0' | '1'

(number) ::= (digit) {(digit)}

(del) ::= (space) | (carriage return)

(delimiter) ::= (del) {(del)}

(printable character) ::= (ASCII 20 hex ... 7E hex)

(control character)::= (ASCII 00 hex ... 1F hex)
| (ASCII 7F hex)

(STX) ::= (ASCII 02 hex)
(ETX) ::= (ASCII 03 hex)
(carriage return) ::= (ASCII 0D hex)
(line feed) ::= (ASCII 0A hex)
(space) ::= (ASCII 20 hex) | ' '

(valid character) ::= (printable character)
| (carriage return) | (line feed)

(field character) ::= (ASCII 20 hex ... 2A hex)
| (ASCII 2C hex ... 7E hex)
| (carriage return) | (line feed)

# SECTION A.2

# TRANSMISSION PROTOCOL

The transmission consists of a start-of-text (STX) character, the various fields, an end-of-text (ETX) character, and a transmission check-sum. The character set consists of the printable ASCII characters and four control characters (STX, ETX, CR, LF). The other control characters should not be used because they can produce undesirable side-effects in the receiving equipment.

**BNF Syntax**

(format) ::= (STX) (design spec) {(field)} (ETX)
(xmit check-sum)

## A.2.1 DESIGN SPEC

The design specification is the first field in the format and doesn't have an identifier. An asterisk terminates the field. This field must be included. The design specification should consist of:

1. User's name and company
2. Date, part number, and revision
3. Manufacturer's device number
4. Other information

**BNF Syntax**

(design spec) ::= {(field character)} '*'

## A.2.2 TRANSMISSION CHECK-SUM

The transmission check-sum is the 16-bit sum (i.e., modulo 65,535) of all ASCII characters transmitted between and including the STX and ETX. The parity bit is excluded in the calculation.

**BNF Syntax**

(xmit check-sum) ::= (hex-digit):4

## A.2.3 FIELDS

Each field starts with a single character identifier. Multiple character identifiers could be used to create sub-fields (i.e., "A1", "A$", or "AB3"). The field is terminated with an asterisk and therefore asterisks cannot be imbedded within the field. While not required, carriage returns and line feeds should be used to improve the readability of the format.

**BNF Syntax**

(field) ::= [(Delimiter)] (field identifier)
{(field character)} '*'

(field identifier) ::= 'C' | 'D' | 'F' | 'G' | 'L' | 'M' | 'P' | 'Q'
| 'R' | 'S' | 'T' | 'V'

(reserved identifier) ::= 'A' | 'B' | 'E' | 'H' | 'I' | 'J' | 'K'
| 'N' | 'O' | 'U' | 'W' | 'X' | 'Y' | 'Z'

**Identifiers**

| | |
|---|---|
| A - * | N - * |
| B - * | O - * |
| C - Check sum | P - Pin sequence |
| D - Device type | Q - Value |
| E - * | R - Resulting vector |
| F - Default fuse state | S - Starting vector |
| G - Security fuse | T - Test cycles |
| H - * | U - * |
| I - * | V - Test vector |
| J - * | W - * |
| K - * | X - * |
| L - Fuse list | Y - * |
| M - Option | Z - * |

* Reserved for future use

SAMPLE TRANSMISSION:

```
(STX)
Acme Logic Design          Joan Engineer          Feb. 29 1983
Widget Decode 756-AB-3456 Rev C  Device Mullard  12AX7*
QP20* QF384* G1*
L0000  1111111011 1111111111 1111000000 0000000000
       0000000000 0000000000 0000000000 0000000000
       0000000000 0000000101 1111111111 1111111111
       0000000000 0000000000 0000111101 1111111111
       1111111111 1111110111 1111111111 1111111111*

L0200  1110101111 1111110000 0000000000 0000000000
       1111111111 1111011011 1111111111 1111111110
       0111111111 1111111111 1111111110 1111111111
       1111111111 1111101111 1111111111 1111101111
       0000000000 0000000000 0000*       C1BB9*

V0001  XXXXXXXXXNXXXXXXXH0N* V0002  XXXXXXXXXNXXXXXXXL1N*
V0003  00XXXXXXXNXXXXXXLXXN* V0004  01XXXXXXXNXXXXXXLXXN*
V0005  10XXXXXXXNXXXXXXLXXN* V0006  11XXXXXXXNXXXXXXHXXN*
V0007  XX00XXXXXNXXXXXLXXXN* V0008  XX01XXXXXNXXXXXHXXXN*
V0009  XX10XXXXXNXXXXXHXXXN* V0010  XX11XXXXXNXXXXXHXXXN*

T02*   S10101010101010101010*   R1ACB678F*
(ETX)  32EB
```

# SECTION A.3

# FORMAT FIELD DEFINITIONS

## A.3.1 DEVICE TO BE PROGRAMMED

The device field defines the specific device being programmed.

**BNF Syntax**

(device) ::= 'D' (family-pinout code) '*'

Example

D9501*

## A.3.2 FUSE INFORMATION

Each device fuse link is assigned a decimal number. The fuse may have two possible states: a zero to specify a low resistance link and a one to specify a high resistance link.

The fuse information is presented in three fields: the default state (F), the fuse list (L), and the fuse check-sum (C). The default state and fuse check-sum fields are optional.

The "F" field is used to define the states of links not explicitly defined in the "L" fields. If the default state is not used, all device links must be defined in the "L" fields.

The "L" field can be any convenient length, and any number of "L" fields can be used. If the state of a link is specified 2 or more times, the last state replaces the preceding entries. The number following the "L" indicates the starting fuse number for the data string.

The link information check-sum is computed by performing a 16 bit addition of 8 bit words constructed from the specified state of each link in the device. The unused bits in the last word are set to zero. The 8 bit words are defined as shown in the following diagram.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| word 00 | msb | | | | | | | lsb |
| Link No. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| word 01 | msb | | | | | | | lsb |
| Link No. | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |

---

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| word 62 | msb | | | | | | | lsb |
| Link No. | - | - | - | - | 499 | 498 | 497 | 496 |

**BNF Syntax**

(fuse information) ::= [(default state)] (fuse list) {(fuse list)} [(fuse check-sum)]

(default state) ::= 'F' (binary-digit) '*'

(fuse list) ::= 'L' (number) (delimiter) {(binary-digit) [(delimiter)]} '*'

(fuse check-sum) ::= 'C' (hex-digit):4 '*'

Example

F0*L0 01010101* L0008 01010111*L1000 0101*CF3BA*

Example

L0200 0110101010101010101110101110101101000100100010010C*

## A.3.3 STRUCTURED FUNCTIONAL TEST INFORMATION

This field specifies one of several test conditions for each pin of a device. A test vector is defined as N test conditions where N is the number of pins on the device.

**Test Conditions**

- 0 - Drive input low
- 1 - Drive input high
- 2-9 - Drive input to supervoltage #2-9
- C - Drive input low, high, low
- K - Drive input high, low, high
- N - Power pins and outputs not tested
- L - Test output low
- H - Test output high
- Z - Test input or output for high impedance
- F - Float input or output

The C and K driving signals are presented after the other inputs are stable. The L, H, and Z tests are performed after all inputs have stabilized including C and K.

The test vectors will be applied to the device under test in numerical order. If any vectors are specified 2 or more times the data in the last vector replaces the previous data. There is no default test vector.

The optional "P" field is used to specify the correspondence of the test conditions to the device pin numbers. If the "P" field is not used, the first condition will be applied to pin 1, the second to pin 2, and so on through pin N.

**BNF Syntax**

(function test) ::= [(pin list)] (test vector) {(test vector)}

(pin list) ::= 'P' (pin number):N '*'

(pin number) ::= (delimiter) (number)

N ::= number of pins on device

(test vector) ::= 'V' (number) (delimiter) (test condition):N '*'

(test condition) ::= (digit) | 'C' | 'F' | 'H' | 'K' | 'L' | 'N' | 'Z'

(reserved condition) ::= 'A' | 'B' | 'D'| 'E' | 'G' | 'I' | 'J' | 'M' | 'O' | 'P' | 'Q' | 'R' | 'S' | 'T' | 'U' | 'V' | 'W' | 'X' | 'Y' | 'Z'

Example

```
P 11  12  13  14  15  16  17  18  19  20
10   9   8   7   6   5   4   3   2   1*

V0001 C01010101NHLLLHHLHLN*
V0002 C01011111NHLLHLLLHLN*
V0003 C10010111NZZZZZZZZZZN*

V4 C01010100NFLHHLFFLLN*
```

## A.3.4 OPTIONAL INFORMATION

Additional option fields may be defined using the letters G, S, R, M, Q, and T. Each field must begin with one of the above letters and be terminated with an asterisk (*). No other restrictions are applied. Therefore, multiple letters could be used to create any number of option fields.

### BNF Syntax

(option field) ::= (option ident.) {(field character)} '*'

(option ident.) ::= 'G' | 'S' | 'R' | 'M' | 'Q' | 'T'

Example

```
Q*
MFG Acme Semiconductor*
M:1234*
```

Data I/O uses five optional fields; three for the Logic Fingerprint™ test, a security fuse field, and a value field.

### LOGIC FINGERPRINT™

The 'S' field defines the starting vector for the Logic Fingerprint™ test. The possible states are 0 (TTL low) and 1 (TTL high). The 'R' field contains the resulting vector or test-sum. The 'T' field denotes the number of test cycles to be run.

### BNF Syntax

(starting vector) ::= 'S' (test condition):N '*'

(resulting vector) ::= 'R' (hex-digit):8 '*'

(test cycles) ::= 'T' (number) '*'

N ::= number of pins on device

Example

```
S0100010000111000111101l0*
R5BCD34A7*
T01*
```

### SECURITY FUSE

The security fuses of certain logic devices may be enabled for programming by sending a "1" in the "G" field.

### BNF Syntax

(security fuse) ::= 'G' (binary-digit) '*'

Example

```
G1*
```

### VALUES

The "Q" field is used to express values or limits required by the receiving device. Two subfields have been defined, the "P" subfield for number of pins on the device and the "F" subfield for the number of fuses. These two values will enable the receiving device process the other fields without knowing the device manufacturer or family/pinout code.

### BNF Syntax

(number of pins) ::= 'QP' (number) '*'

(fuse limit) ::= 'QF' (number) '*'

Example

```
QP24*        QF1024*
```

# SECTION A.4

# OTHER RULES

## A.4.1 TRANSPORTABILITY

All receiving machines should have a "kernel" mode to ignore all optional fields so that the actual programming data will be transportable. For example, as allowed in the format, optional fields can be sent to specify additional checksums. A receiving machine in the "kernel" mode could ignore this information yet receive the link information required to program the device.

If the optional "F" field is used to avoid transmitting link data, transportability could be lost. Therefore, whenever practical, data should be transmitted for all links of the device.

Some computer operating systems add control characters after each line making it very difficult to compute the transmission check-sum. To disable the transmission check-sum, receiving equipment should accept "0000" as a valid check-sum.

### BNF Syntax

(kernel) ::= (STX) (design spec) (min. fuse information) (ETX) (xmit check-sum)

(design spec) ::= {(field character)} '*'

(min. fuse information) ::= (fuse list) {(fuse list)}

Example:

```
(STX)
Acme Logic Design          Jane Engineer          Feb. 29 1983
Widget Decode 756-AB-3456 Rev C Device Mullard 12AX7*

L0000 1111111011 1111111111 1111000000 0000000000
      0000000000 0000000000 0000000000 0000000000
      0000000000 0000000101 1111111111 1111111111
      0000000000 0000000000 0000111101 1111111111
      1111111111 1111110111 1111111111 1111111111*

L0200 1110101111 1111110000 0000000000 0000000000
      1111111111 1111011011 1111111111 1111111110
      0111111111 1111111111 1111111110 1111111111
      1111111111 1111101111 1111111111 1111101111
      0000000000 0000000000 0000*

(ETX)00000
```

## A.4.2 RESTRICTIONS AND VARIATIONS

### VARIATIONS FROM PRESENT JEDEC STANDARD

a) The delimiter after the link number or the vector number may be any combination of carriage returns, line feeds, and/or spaces.

b) The check-sum and sum-check are renamed fuse check-sum and xmit check-sum respectively.

### VARIATIONS FOR 303A-VO1 LOGICPAK™

a) The link number ("L" field) and the vector number ("V" field) must be 4-digit numbers.

b) The delimiter after the link number and the vector number must include a space.

c) Spaces are not allowed between the terminating asterisk of one field and the identifier of the next field.

d) The security fuse ("G" field) requires a space after the binary digit.

e) The kernel mode is not implemented.

f) The "D" field must have a valid family and pinout code.

(device) ::= 'D' (hex-digit):4 '*'

# APPENDIX B

# REFERENCE MATERIAL

## Table B-1. LogicPak™ Family Codes and Pinout Codes

| Device | Family | Pinout | LogicPak™ | P/T Adapter | | Design Adapter | |
|---|---|---|---|---|---|---|---|
| **Advanced Micro Devices** | | | | | | **DATA I/O Part Numbers** | |
| AmPAL 16L8 | 97 | 17 | VO1 | 303A-004 | VO1 | 303A-100 | VO1 |
| AmPAL 16R8 | 97 | 24 | VO1 | 303A-004 | VO1 | 303A-100 | VO1 |
| AmPAL 16R6 | 97 | 24 | VO1 | 303A-004 | VO1 | 303A-100 | VO1 |
| AmPAL 16R4 | 97 | 24 | VO1 | 303A-004 | VO1 | 303A-100 | VO1 |
| AmPAL 16LO8 | 97 | 17 | VO1 | 303A-004 | VO1 | 303A-100 | VO2 |
| AmPAL 16H8 | 97 | 25 | VO1 | 303A-004 | VO1 | 303A-100 | VO2 |
| AmPAL 16HO8 | 97 | 25 | VO1 | 303A-004 | VO1 | 303A-100 | VO2 |
| **Harris Semiconductor** | | | | | | **DATA I/O Part Numbers** | |
| HPL77153/82S153 | 98 | 04 | VO1 | 303A-003 | VO1 | 303A-101 | VO1 |
| HPL77209/16L8 | 98 | 01 | VO1 | 303A-003 | VO1 | 303A-100 | VO1 |
| HPL77210/16R4 | 98 | 02 | VO1 | 303A-003 | VO1 | 303A-100 | VO1 |
| HPL77211/16R6 | 98 | 02 | VO1 | 303A-003 | VO1 | 303A-100 | VO1 |
| HPL77212/16R8 | 98 | 02 | VO1 | 303A-003 | VO1 | 303A-100 | VO1 |
| HPL77215/16H8 | 98 | 01 | VO1 | 303A-003 | VO1 | * | * |
| HPL77216/16P8 | 98 | 03 | VO1 | 303A-003 | VO1 | * | * |
| **Monolithic Memories** | | | | | | **DATA I/O Part Numbers** | |
| PAL 10H8 | 22 | 18 | VO1 | 303A-002 | VO2 | 303A-100 | VO1 |
| PAL 12H6 | 22 | 19 | VO1 | 303A-002 | VO2 | 303A-100 | VO1 |
| PAL 14H4 | 22 | 20 | VO1 | 303A-002 | VO2 | 303A-100 | VO1 |
| PAL 16H2 | 22 | 22 | VO1 | 303A-002 | VO2 | 303A-100 | VO1 |
| PAL 10L8 | 22 | 13 | VO1 | 303A-002 | VO2 | 303A-100 | VO1 |
| PAL 12L6 | 22 | 14 | VO1 | 303A-002 | VO2 | 303A-100 | VO1 |
| PAL 14L4 | 22 | 15 | VO1 | 303A-002 | VO2 | 303A-100 | VO1 |
| PAL 16L2 | 22 | 16 | VO1 | 303A-002 | VO2 | 303A-100 | VO1 |
| PAL 16A4 | 22 | 24 | VO1 | 303A-002 | VO2 | 303A-100 | VO1 |
| PAL 16X4 | 22 | 24 | VO1 | 303A-002 | VO2 | 303A-100 | VO1 |
| PAL 16R4/16R4A | 22 | 24 | VO1 | 303A-002 | VO2 | 303A-100 | VO1 |
| PAL 16R6/16R6A | 22 | 24 | VO1 | 303A-002 | VO2 | 303A-100 | VO1 |
| PAL 16L8/16L8A | 22 | 17 | VO1 | 303A-002 | VO2 | 303A-100 | VO1 |
| PAL 16R8/16R8A | 22 | 24 | VO1 | 303A-002 | VO2 | 303A-100 | VO1 |
| PAL 16C1 | 22 | 21 | VO1 | 303A-002 | VO2 | 303A-100 | VO1 |
| PAL 12H10 | 22 | 07 | VO1 | 303A-002 | VO2 | 303A-100 | VO1 |
| PAL 14H8 | 22 | 08 | VO1 | 303A-002 | VO2 | 303A-100 | VO1 |
| PAL 16H6 | 22 | 09 | VO1 | 303A-002 | VO2 | 303A-100 | VO1 |
| PAL 18H4 | 22 | 10 | VO1 | 303A-002 | VO2 | 303A-100 | VO1 |
| PAL 20H2 | 22 | 11 | VO1 | 303A-002 | VO2 | 303A-100 | VO1 |
| PAL 12L10 | 22 | 01 | VO1 | 303A-002 | VO2 | 303A-100 | VO1 |
| PAL 14L8 | 22 | 02 | VO1 | 303A-002 | VO2 | 303A-100 | VO1 |
| PAL 16L6 | 22 | 03 | VO1 | 303A-002 | VO2 | 303A-100 | VO1 |
| PAL 18L4 | 22 | 04 | VO1 | 303A-002 | VO2 | 303A-100 | VO1 |
| PAL 20L2 | 22 | 06 | VO1 | 303A-002 | VO2 | 303A-100 | VO1 |
| PAL 20C1 | 22 | 12 | VO1 | 303A-002 | VO2 | 303A-100 | VO1 |
| PAL 20L10 | 22 | 06 | VO1 | 303A-002 | VO2 | 303A-100 | VO1 |
| PAL 20X10 | 22 | 23 | VO1 | 303A-002 | VO2 | 303A-100 | VO1 |
| PAL 20X8 | 22 | 23 | VO1 | 303A-002 | VO2 | 303A-100 | VO1 |
| PAL 20X4 | 22 | 23 | VO1 | 303A-002 | VO2 | 303A-100 | VO1 |
| PAL 20L8 | 22 | 26 | VO1 | 303A-002 | VO2 | 303A-100 | VO1 |
| PAL 20R8 | 22 | 27 | VO1 | 303A-002 | VO2 | 303A-100 | VO1 |
| PAL 20R6 | 22 | 27 | VO1 | 303A-002 | VO2 | 303A-100 | VO1 |
| PAL 20R4 | 22 | 27 | VO1 | 303A-002 | VO2 | 303A-100 | VO1 |

* under development

| Device | Family | Pinout | LogicPak™ | P/T Adapter | | Design Adapter | |
|---|---|---|---|---|---|---|---|
| **National Semiconductor** | | | | | | **DATA I/O Part Numbers** | |
| PAL 10H8 | 95 | 18 | VO1 | 303A-002 | VO1 | 303A-100 | VO1 |
| PAL 12H6 | 95 | 19 | VO1 | 303A-002 | VO1 | 303A-100 | VO1 |
| PAL 14H4 | 95 | 20 | VO1 | 303A-002 | VO1 | 303A-100 | VO1 |
| PAL 16H2 | 95 | 22 | VO1 | 303A-002 | VO1 | 303A-100 | VO1 |
| PAL 10L8 | 95 | 13 | VO1 | 303A-002 | VO1 | 303A-100 | VO1 |
| PAL 12L6 | 95 | 14 | VO1 | 303A-002 | VO1 | 303A-100 | VO1 |
| PAL 14L4 | 95 | 15 | VO1 | 303A-002 | VO1 | 303A-100 | VO1 |
| PAL 16L2 | 95 | 16 | VO1 | 303A-002 | VO1 | 303A-100 | VO1 |
| PAL 16R4 | 95 | 24 | VO1 | 303A-002 | VO1 | 303A-100 | VO1 |
| PAL 16R6 | 95 | 24 | VO1 | 303A-002 | VO1 | 303A-100 | VO1 |
| PAL 16L8 | 95 | 17 | VO1 | 303A-002 | VO1 | 303A-100 | VO1 |
| PAL 16R8 | 95 | 24 | VO1 | 303A-002 | VO1 | 303A-100 | VO1 |
| PAL 16C1 | 95 | 21 | VO1 | 303A-002 | VO1 | 303A-100 | VO1 |
| PAL 20L10 | 95 | 06 | VO1 | 303A-002 | VO1 | 303A-100 | VO1 |
| PAL 20X10 | 95 | 23 | VO1 | 303A-002 | VO1 | 303A-100 | VO1 |
| PAL 20X8 | 95 | 23 | VO1 | 303A-002 | VO1 | 303A-100 | VO1 |
| PAL 20X4 | 95 | 23 | VO1 | 303A-002 | VO1 | 303A-100 | VO1 |
| **Signetics** | | | | | | **DATA I/O Part Numbers** | |
| FPLA 82S101/100 | 96 | 01 | VO1 | 303A-001 | VO1 | 303A-101 | VO1 |
| FPLS 82S104/105 | 96 | 03 | VO1 | 303A-001 | VO1 | 303A-101 | VO1 |
| FPRP 82S106/107 | 96 | 04 | VO1 | 303A-001 | VO1 | 303A-101 | VO1 |
| FPGA 82S102/103 | 96 | 02 | VO1 | 303A-001 | VO1 | 303A-101 | VO1 |
| FPLA 82S152/153 | 96 | 05 | VO1 | 303A-001 | VO1 | 303A-101 | VO1 |
| FPLS 82S158/159 | * | * | * | * | * | * | * |
| **Texas Instruments** | | | | | | **DATA I/O Part Numbers** | |
| PAL 16L8 | 99 | 17 | VO1 | 303A-006 | VO1 | 303A-100 | VO1 |
| PAL 16R4 | 99 | 24 | VO1 | 303A-006 | VO1 | 303A-100 | VO1 |
| PAL 16R6 | 99 | 24 | VO1 | 303A-006 | VO1 | 303A-100 | VO1 |
| PAL 16R8 | 99 | 24 | VO1 | 303A-006 | VO1 | 303A-100 | VO1 |
| FPLA 74FP840/839 | * | * | * | * | * | * | * |

*under development

## Glossary

**address field.** Optional set of control characters in a data translation format. It defines the address of the next data byte.

**address offset.** A four-digit hexadecimal value subtracted from addresses on input and added to addresses output from the programmer. The result is added to the begin device address or begin RAM address as applicable.

**begin device address.** The first device address from which or to which data are being transferred.

**begin RAM address.** The first address of the programmer data RAM from which or to which data are to be transferred.

**blank check.** A test performed by a programmer to detect the presence of any programmed bits. A device with no programmed bits is "blank."

**block size.** The hexadecimal number of bytes to be transferred in a data transfer.

**configuration number.** A four-digit hexadecimal number that identifies the software revision level of the programmer.

**data translation format.** Form in which the translator software accepts input data. Also the form for data output by the unit.

**default value.** The value the unit uses for a parameter unless the operator specifies another value.

**device.** Any PROM, EPROM, MOS PROM, or programmable logic array.

**end code.** Character in a data translation format that signals the completion of a data transfer.

**error code.** A code that signals specific errors to the operator.

**family and pinout codes.** Two-digit codes used by some Data I/O programming modules to identify programming variables including pinout, address limit and programming algorithms.

**full duplex.** Simultaneous, two-way independent transmission in both directions.

**handshaking.** The required sequence of signals for communication between two units. The I/O bus protocol for a unit defines its handshaking requirements. This is especially true for asynchronous I/O systems in which each signal requires a response to complete an I/O operation.

**half duplex.** Transmission alternately in either direction but not both directions simultaneously.

**illegal-bit test.** A test performed by a programmer to detect the presence of any programmed bits of incorrect polarity (illegal bits).

**logic gate.** Digital circuits with two or more logic inputs and one output, with a logic level determined by the logic levels present at the inputs. The basic logic gates are:

**OR** produces a 1 (hi) when one or more inputs are 1.
**AND** produces a 1 only when all inputs are 1.
**NOR** produces Ø (low) when any input is 1.
**NAND** produces a Ø only when all inputs are at different levels
**EX-NOR** produces a 1 only when the inputs are at the same logic level.

**mode.** A software routine in a machine, characterized by a specific automatic sequence of steps.

**select function** (also select code). A two-digit hexadecimal number used to specify data translation formats, serial interface operations, or certain RAM data manipulations.

**start code.** Character in a data translation format that signals the beginning of a data transfer.

**sum-check.** A summation of bits calculated according to the rules of simple addition and usually expressed as a four-digit hexadecimal number; any carry from the most significant bit or digit is discarded. A sum-check is used to verify the integrity of data transfers.

**waveforms (programmable).** The graphical representation of the timing and magnitude of programming pulses. If the programming waveforms are not kept within tolerance, programming yield is jeopardized.

## Abbreviations

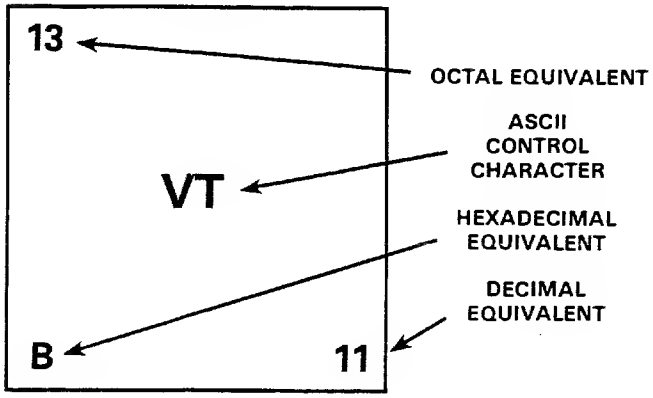
The following is a list of abbreviations commonly used in Data I/O Instruction manuals.

amp - amplifier or ampere
ASCII - American Standard Code for Information Interchange
CE - chip enable
CMOS - complementary metal oxide semiconductor
CP - clock pulse
CR - carriage return
CTRL - control
CTS - clear to send
DAC - digital-to-analog converter
dV/dt - delta voltage versus delta time
E - enable pulse derived from the 6802 microprocessor
EPROM - erasable, programmable read-only memory
ESC - escape
ETX - end of text
FPGA - field programmable gate array
FPLA - field programmable logic array
FPLS - field programmable logic sequencer
H&L - high and low
hex - hexadecimal
IFL - integrated fuse logic
I/O - input/output
JEDEC - Joint Electron Device Engineering Council
K - thousand
LED - light-emitting diode
LF - line feed
MHz - megahertz
No. - number
op-amp - operational amplifier
PAL - programmable array logic
PALASM - PAL assembler
PCS - program card set
PLDS - programmable logic development system
PROM - programmable read-only memory
P/T - programming/testing
RAM - random-access memory
Rev - revision
R/$\overline{W}$ - read/$\overline{write}$
S/L - control line
SO - serial output
STX - start of text
SYNC - synchronous
TP - test point
TTL - transistor-transistor logic
V•Ø2 - timing pulse, sum of E and VMA
V - volt
$V_{CC}$ - power supply voltage applied to integrated circuits
VMA - valid memory address

**ASCII CONTROL CHARACTERS**

| | |
|---|---|
| ACK | acknowledge |
| BEL | bell |
| BS | backspace |
| CAN | cancel |
| CR | carriage return |
| DC1 | playback on, CNTL Q, X-ON |
| DC2 | record on, CNTL R, PUNCH-ON, SOM |
| DC3 | playback off, CNTL S, X-OFF |
| DC4 | record off, CNTL T, PUNCH-OFF, EOM |
| DEL | delete, rubout |
| DLE | data link escape |
| EM | end of medium |
| ENQ | enquiry |
| EOT | end of transmission |
| ESC | escape |
| ETB | end of transmission block |
| ETX | end of text |
| FF | form feed |
| FS | file separater |
| GS | group separater |
| HT | horizontal tabulation |
| LF | line feed |
| NAK | negative acknowledge |
| NUL | null |
| RS | record separater |
| SI | shift in |
| SO | shift out |
| SOH | start of heading |
| STX | start of text |
| SUB | substitute |
| SYN | synchronous idle |
| US | unit separater |
| VT | vertical tab |

**Code Chart Key**



## Cross-Reference Chart of Number Bases

| Binary | Octal | Hexadecimal | Decimal | Standard Abbreviation |
|---|---|---|---|---|
| 0000 | 0 | 0 | 0 | |
| 0001 | 1 | 1 | 1 | |
| 0010 | 2 | 2 | 2 | |
| 0011 | 3 | 3 | 3 | |
| 0100 | 4 | 4 | 4 | |
| 0101 | 5 | 5 | 5 | |
| 0110 | 6 | 6 | 6 | |
| 0111 | 7 | 7 | 7 | |
| 1000 | 10 | 8 | 8 | |
| 1001 | 11 | 9 | 9 | |
| 1010 | 12 | A | 10 | |
| 1011 | 13 | B | 11 | |
| 1100 | 14 | C | 12 | |
| 1101 | 15 | D | 13 | |
| 1110 | 16 | E | 14 | |
| 1111 | 17 | F | 15 | |
| 0001 0000 | 20 | 10 | 16 | |
| 0010 0000 | 40 | 20 | 32 | |
| 0100 0000 | 100 | 40 | 64 | |
| 1000 0000 | 200 | 80 | 128 | |
| 0001 0000 0000 | 400 | 100 | 256 | |
| 0010 0000 0000 | 1000 | 200 | 512 | |
| 0100 0000 0000 | 2000 | 400 | 1,024 | 1K |
| 1000 0000 0000 | 4000 | 800 | 2,048 | 2K |
| 1100 0000 0000 | 6000 | C00 | 3,072 | 3K |
| 0001 0000 0000 0000 | 10000 | 1000 | 4,096 | 4K |
| 0001 0100 0000 0000 | 12000 | 1400 | 5,120 | 5K |
| 0001 1000 0000 0000 | 14000 | 1800 | 6,144 | 6K |
| 0001 1100 0000 0000 | 16000 | 1C00 | 7,168 | 7K |
| 0010 0000 0000 0000 | 20000 | 2000 | 8,192 | 8K |
| 0010 0100 0000 0000 | 22000 | 2400 | 9,216 | 9K |
| 0010 1000 0000 0000 | 24000 | 2800 | 10,240 | 10K |
| 0100 0000 0000 0000 | 40000 | 4000 | 16,384 | 16K |
| 1000 0000 0000 0000 | 100000 | 8000 | 32,768 | 32K |
| 0001 0000 0000 0000 0000 | 200000 | 10000 | 65,536 | 64K |

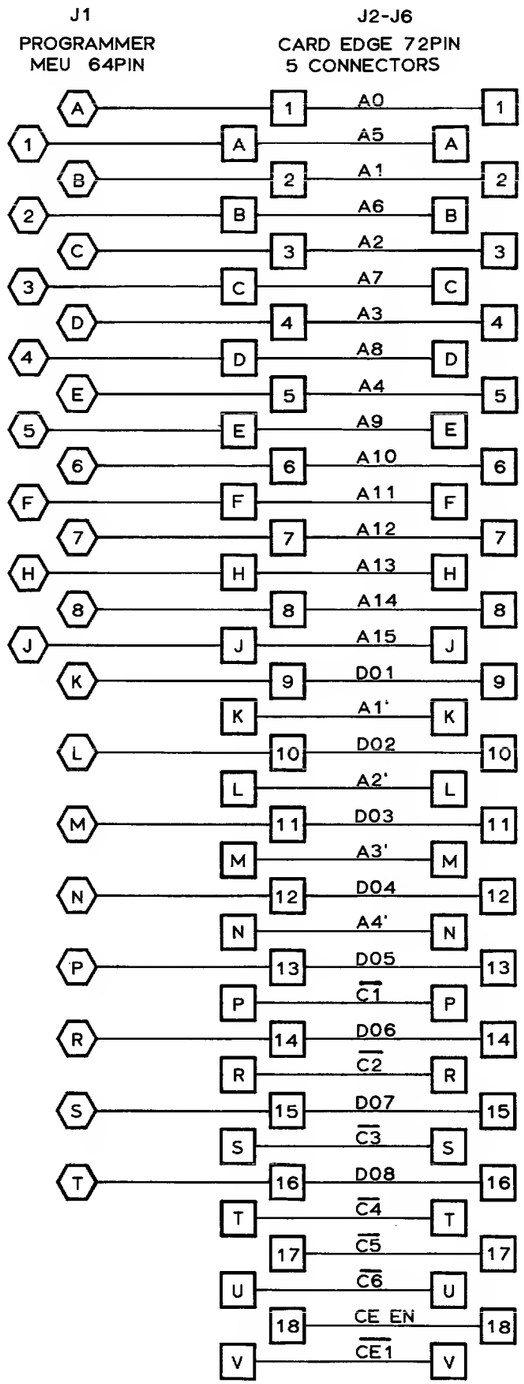
## ASCII and IEEE Code Chart

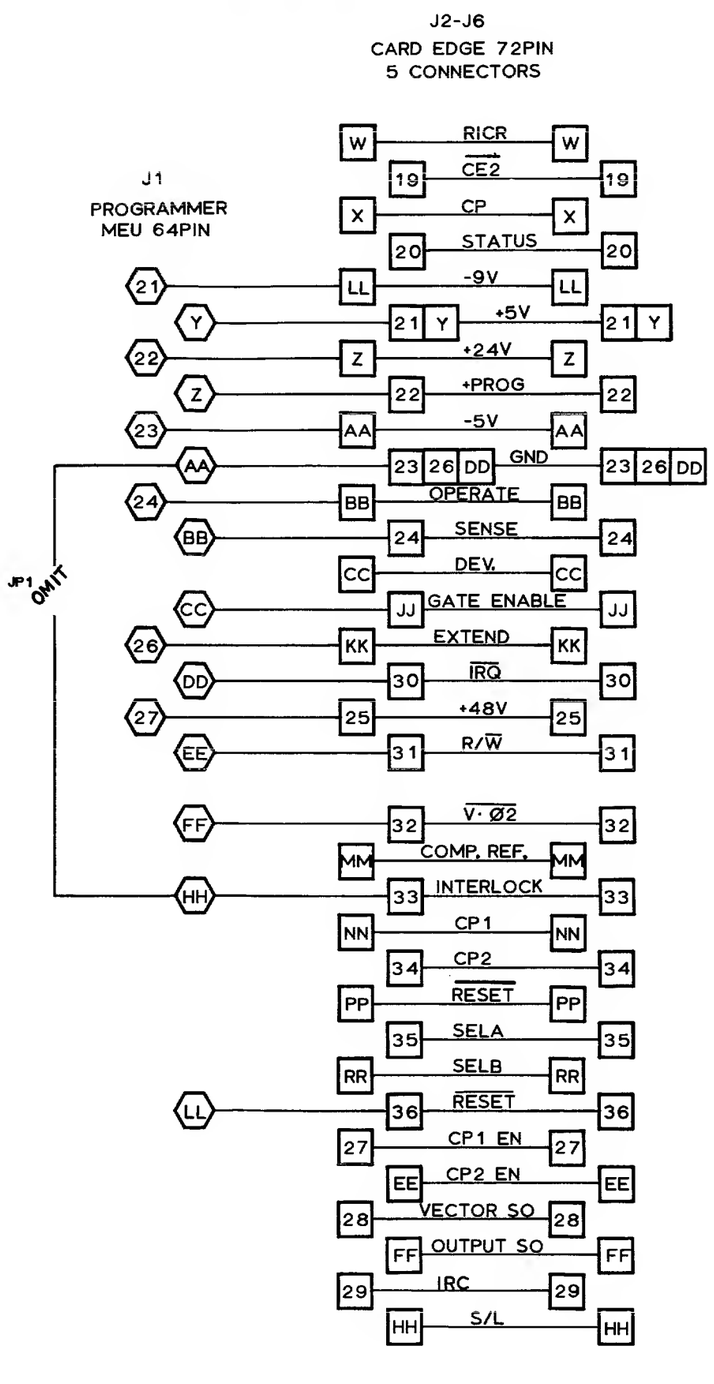
| BITS 4 3 2 1 | 7 6 5 = 0 0 0 | 0 0 1 | 0 1 0 | 0 1 1 | 1 0 0 | 1 0 1 | 1 1 0 | 1 1 1 |
|---|---|---|---|---|---|---|---|---|
| | CONTROL | | NUMBERS AND SYMBOLS | | UPPERCASE | | LOWERCASE | |
| 0 0 0 0 | 0 NUL 0 0 | 20 DLE 10 16 | 40 SP 20 32 | 60 0 30 48 | 100 @ 40 64 | 120 P 50 80 | 140 ` 60 96 | 160 p 70 112 |
| 0 0 0 1 | 1 SOH 1 1 | 21 DC1 11 17 | 41 ! 21 33 | 61 1 31 49 | 101 A 41 65 | 121 Q 51 81 | 141 a 61 97 | 161 q 71 113 |
| 0 0 1 0 | 2 STX 2 2 | 22 DC2 12 18 | 42 " 22 34 | 62 2 32 50 | 102 B 42 66 | 122 R 52 82 | 142 b 62 98 | 162 r 72 114 |
| 0 0 1 1 | 3 ETX 3 3 | 23 DC3 13 19 | 43 # 23 35 | 63 3 33 51 | 103 C 43 67 | 123 S 53 83 | 143 c 63 99 | 163 s 73 115 |
| 0 1 0 0 | 4 EOT 4 4 | 24 DC4 14 20 | 44 $ 24 36 | 64 4 34 52 | 104 D 44 68 | 124 T 54 84 | 144 d 64 100 | 164 t 74 116 |
| 0 1 0 1 | 5 ENQ 5 5 | 25 NAK 15 21 | 45 % 25 37 | 65 5 35 53 | 105 E 45 69 | 125 U 55 85 | 145 e 65 101 | 165 u 75 117 |
| 0 1 1 0 | 6 ACK 6 6 | 26 SYN 16 22 | 46 & 26 38 | 66 6 36 54 | 106 F 46 70 | 126 V 56 86 | 146 f 66 102 | 166 v 76 118 |
| 0 1 1 1 | 7 BEL 7 7 | 27 ETB 17 23 | 47 ' 27 39 | 67 7 37 55 | 107 G 47 71 | 127 W 57 87 | 147 g 67 103 | 167 w 77 119 |
| 1 0 0 0 | 10 BS 8 8 | 30 CAN 18 24 | 50 ( 28 40 | 70 8 38 56 | 110 H 48 72 | 130 X 58 88 | 150 h 68 104 | 170 x 78 120 |
| 1 0 0 1 | 11 HT 9 9 | 31 EM 19 25 | 51 ) 29 41 | 71 9 39 57 | 111 I 49 73 | 131 Y 59 89 | 151 i 69 105 | 171 y 79 121 |
| 1 0 1 0 | 12 LF A 10 | 32 SUB 1A 26 | 52 * 2A 42 | 72 : 3A 58 | 112 J 4A 74 | 132 Z 5A 90 | 152 j 6A 106 | 172 z 7A 122 |
| 1 0 1 1 | 13 VT B 11 | 33 ESC 1B 27 | 53 + 2B 43 | 73 ; 3B 59 | 113 K 4B 75 | 133 [ 5B 91 | 153 k 6B 107 | 173 { 7B 123 |
| 1 1 0 0 | 14 FF C 12 | 34 FS 1C 28 | 54 , 2C 44 | 74 V 3C 60 | 114 L 4C 76 | 134 \ 5C 92 | 154 l 6C 108 | 174 \| 7C 124 |
| 1 1 0 1 | 15 CR D 13 | 35 GS 1D 29 | 55 - 2D 45 | 75 = 3D 61 | 115 M 4D 77 | 135 ] 5D 93 | 155 m 6D 109 | 175 } 7D 125 |
| 1 1 1 0 | 16 SO E 14 | 36 RS 1E 30 | 56 . 2E 46 | 76 > 3E 62 | 116 N 4E 78 | 136 ^ 5E 94 | 156 n 6E 110 | 176 ~ 7E 126 |
| 1 1 1 1 | 17 SI F 15 | 37 US 1F 31 | 57 / 2F 47 | 77 ? 3F 63 | 117 O 4F 79 | 137 ← 5F 95 | 157 o 6F 111 | 177 Rubout 7F 127 |
| | Addressed Commands | Universal Commands | Listen Addresses | | Talk Addresses | | Secondary Addresses or Commands | |

## SCHEMATICS

| | |
|---|---|
| 30-702-1938 | Motherboard |
| 30-701-1939 | Waveform Generator |
| 30-701-1940 | Sink Driver |
| 30-701-1941 | T/Rise Comparator |
| 30-701-1942 | Controller/Memory |
| 30-702-1943 | Pin Driver |
| 30-701-1944 | Extender Board |

| REVISIONS | | | | | |
|---|---|---|---|---|---|
| LTR | DESCRIPTION | DR | CHK | APPR'D | DATE |
| A | RELEASE | JK | | | [illegible] |
| B | ECN 4691 | CL | | | |
| C | ECN #4785 3-83 | SH | [illegible] | 69 | 3/83 |




NOTES: UNLESS OTHERWISE SPECIFIED

1. LAST REFERENCE DESIGNATOR USED: JP1, J6

| APPROVALS: | | UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES. | DATA I/O | |
|---|---|---|---|---|
| DSN. ENGR. | DATE 4-12-82 | TOLERANCES, UNLESS OTHERWISE SPECIFIED: .XX ± .XXX ± ANGULAR | TITLE SCHEMATIC DIAGRAM, MOTHER BOARD | |
| MFG. ENGR. | DATE 11/24/82 | | | |
| QUAL. ASSUR. | DATE | DRAWN BY: JAEHEE KIM | DATE 12/81 | SIZE D / CODE INDENT. NO. 54193 / DRAWING NO. 30-702-1938 |
| ENGR. MGR. | DATE 11/18/82 | CHECKED BY: | DATE | SCALE — / 12-17-81 / SHEET 1OF1 |

| REVISIONS | | | | | |
|---|---|---|---|---|---|
| LTR | DESCRIPTION | DR | CHK | APPR'D | DATE |
| A | RELEASE | MP | 7/16 | | 7/12/82 |
| B | ECN #4785 3-83 | SH | TW | GT | 3/83 |

VCC SUPPLY DAC

CE SUPPLY DAC

BIT SUPPLY DAC

CONTROL REG

DATA LATCH

VCC SUPPLY

CE SUPPLY

BIT SUPPLY

DAC REF

CURRENT SENSE INTEGRATOR

RESET MEMORY

POWER ON RESET

SCOPE SYNC

SEL B SEL A

RESET

COMP REF

VFE CLAMP

VPUP

VCC

VCC SENSE

CE SUPPLY

BIT SUPPLY

GND

701-1939-001 SHOWN, EXCEPT AS NOTED.

NOTES: UNLESS OTHERWISE SPECIFIED.

1. ALL RESISTORS ARE IN OHMS, 1/4W, 5%.
2. ALL CAPACITORS ARE IN MICROFARADS, 50V.
3. LAST REF DES USED.
   RP2, U15, C31, Q8, CR12, R43, TP6, VR1.
4. POWER AND GROUND.

| REF. DES. | +5V | GND | +24V | -9V |
|---|---|---|---|---|
| U14,15 | 14 | 7 | | |
| U11,12 | 20 | 10 | | |
| U13 | 18 | 8 | | |
| U6 | | | 4 | 11 |

5. ALL DIODES ARE 1N4148.
6. -002 SHOWN.

| APPROVALS: | | UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES. | | DATA I/O | |
|---|---|---|---|---|---|
| DSN. ENGR. | DATE 4-14-82 | TOLERANCES, UNLESS OTHERWISE SPECIFIED: .XX ± .XXX ± ANGULAR | | TITLE SCHEMATIC DIAGRAM, WAVE FORM GENERATOR | |
| MFG. ENGR. | DATE 7/16/82 | | | | |
| QUAL. ASSUR. | DATE 7/20/82 | DRAWN BY: M. Patten | DATE 5/12/82 | SIZE D | CODE INDENT. NO. 54193 |
| ENGR. MGR. | DATE 7/20/82 | CHECKED BY: Dan Cocca | DATE 7/20 | SCALE | SHEET 1 OF 1 |

DRAWING 49-701

| REVISIONS | | | | | |
|---|---|---|---|---|---|
| LTR | DESCRIPTION | DR | CHK | APPR'D | DATE |
| A | RELEASE | S.H. | | | 6-30 |
| B | ECN #4785 3-83 | SH | | | |

INPUT VECTOR SHIFT REGISTER

SINK DRIVERS

PARITY GEN

RECIRCULATE GATING

SECONDARY DECODE

CP ANDCE GATE

NOTES: UNLESS OTHERWISE SPECIFIED
1. ALL RESISTORS ARE 2.2K OHMS, 1/4W, 5%.
2. ALL CAPACITORS ARE IN MICROFARADS.
3. LAST REFERENCE DESIGNATOR USED: U23, C9, RP4, RI2, J2
4.

| IC | +5V | GND |
|---|---|---|
| U1-3,5,7,8,10,12-22 | 14 | 7 |
| U23 | 16 | 8 |

UNUSED GATES:

U3 7406 U3 7406

U1 74LS00 2 PLACES

| APPROVALS: | | UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES | DATA I/O | |
|---|---|---|---|---|
| DSN ENGR | DATE 11/1/82 | TOLERANCES UNLESS OTHERWISE SPECIFIED XX ± XXX ± ANGULAR | TITLE SCHEMATIC DIAGRAM, SINK DRIVER | |
| MFG ENGR | DATE 11/4/82 | | | |
| QUAL. ASSUR. | DATE JUL 12 '84 | DRAWN BY JAEHEE KIM | DATE 12/81 | SIZE D |
| ENGR MGR. | DATE 11/1/82 | CHECKED BY | DATE 11/1/82 | CODE INDENT NO 54193 |
| | | | | DRAWING NO 30-701-1940 |
| | | | | SCALE — SHEET 1OF1 |

| REVISIONS | | | | | |
|---|---|---|---|---|---|
| LTR | DESCRIPTION | DR | CHK | APPR'D | DATE |
| A | RELEASE | MP | | GS | 6-30 |
| B | ECN 4677 | JK | MH | | 10-20 |
| C | ECN 4765 | | | | |
| D | ECN #4785 3-83 | SH | MH | GS | 3/83 |

BIT SUPPLY (1,D3)

+48V R1 5K,25T CW, R2 1K

+5V CR1 IN747A 3.6V, R7 1K, R8 39 — CURRENT SOURCE

Q4 2N2484, R3 32.4K 1%

U1 MPQ3799 — Q3 MJE800 — BIT SW#1

R4 1K — U1 MPQ3799 — Q2 MJE800 — BIT SW#2

R5 1K — U1 MPQ3799 — Q1 MJE800 — CE SW

BIT SUPPLY (1,D5)

J1
- 18 CE SUPPLY
- 20 BIT SUPPLY
- 34 CE SW
- 25 BIT SW#2
- 33 BIT SW#1

P1
- DO1 9
- DO2 10
- DO3 11
- DO4 12
- DO5 13
- DO6 14
- DO7 15
- DO8 16
- C5 17
- RESET PP
- OUTPUT SO FF → OUTPUT SO (2,A4)
- S/L HH → S/L (2,A4)
- CP X → CP (2,A4)
- C4 T

SW CONTROL REG.

U14 74LS273 (D, Q, CK, CL)

U3 74LS38 — U2 MPQ3725 — R8 330, C1 150PF — BIT SW#1 TR CONTROL

U3 74LS38 — U2 MPQ3725 — R10 330, C5 150PF — BIT SW#2 TR CONTROL

U3 74LS38 — U2 MPQ3725 — R9 330, C8 150PF — CE SW TR CONTROL

U13 MPQ3725 — R28 4.7K, C2 .0068/50V, C3 .001, C4 .0022/50V, R27 4.7K

U2 MPQ3725 — R26 4.7K, C6 .001, C7 .0022/50V

U13 MPQ3725 — R24 4.7K, C11 .0068/50V, C9 .001, C10 .0022/50V, R23 4.7K

+5V — R34,1K,1/2W; R32,1K,1/2W; R33,1K,1/2W

VCC & SW COMPARATORS

+5V — R12, R11, R13, R14 3.9K (4)

U15 74LS244 (EN OUT, EN IN)

U4 339

R16 100K 1%; R18 33.2K 1%; R22 33.2K 1%; R20 33.2K 1%

R15 100K 1%; R17 100K 1%; R21 100K 1%; R19 100K 1%

21 VCC SENSE

30, 32 GROUND; 31

+5V — Y — 21 — C20 22/15V; C16,17 .01/50V; C19 1/50V; C12,13,15 .01/50V; C18 .1; C21 .01

+PROG 22

+48V 25

GND 23, 26 — DD

STATUS 20

COMP REF MM → COMP REF (2,A8)

C14 .01/50V

STATUS GATE

R25 3.9K, R29 3.9K

JP2, JP1 — JUMPER OPT

NOTES: UNLESS OTHERWISE SPECIFIED.
1. ALL RESISTORS ARE IN OHMS 1/4W, 5%.
2. ALL CAPACITORS ARE IN MFD, 100V.
3. LAST REF DES USED R34, C21, JP2, Q7, CR1, U21, J2
4. POWER AND GROUND.

| REF DES | +5 | GND | +PROG |
|---|---|---|---|
| U3 | 14 | 7 | |
| U4-U12 | | 12 | 3 |
| U14, 15 | 20 | 10 | |
| U16,18,21,19 | 18 | 8 | |

| APPROVALS: | | UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES | DATA I/O | |
|---|---|---|---|---|
| DSN. ENGR. | DATE | TOLERANCES, UNLESS OTHERWISE SPECIFIED: XX ± XXX ± ANGULAR | TITLE SCHEMATIC DIAGRAM, T/RISE COMPARATOR | |
| MFG. ENGR. | DATE 7/12/82 | | | |
| QUAL. ASSUR. | DATE 8/12/82 | DRAWN BY: M. Potter | DATE 8/23/82 | SIZE D; CODE IDENT. NO. 54193; DRAWING NO. 30-701-1941 |
| ENGR. MGR. | DATE 11/19/82 | CHECKED BY: | DATE 9/14/82 | SCALE ——; SHEET 1OF2 |

| REVISIONS | | | | | |
|---|---|---|---|---|---|
| LTR | DESCRIPTION | DR | CHK | APPR'D | DATE |
| A | SEE SHT 1 | MT | | GS | 6-30 |
| B | SEE SHT 1 | JK | | | 10-20 |
| C | SEE SHT 1 | MP | | | 2-4-3 |
| D | SEE SHT1 3-83 | SH | MW | GS | 3/23 |

COMPARATORS

OUTPUT SHIFT REG.

U12 339 · U11 339 · U10 339 · U9 339 · U5 339 · U6 339 · U7 339 · U8 339

U17 3.9K (R30 3.9K, R31 3.9K)

U16 74LS166 · U18 74LS166 · U21 74LS166 · U19 74LS166

U20 3.9K

+5V

J1

I/O 1 SINK/SOURCE

2 2

3 3

4 4

5 5

6 6

7 7

8 8

9 12

10 11

11 10

12 9

13 15

14 14

15 13

16 16

17 46

18 45

19 44

20 43

21 50

22 49

23 48

24 47 24 SINK/SOURCE

25 29 25 SINK

26 28 26 SINK

27 37 27 SINK/SOURCE

28 38 28 SINK/SOURCE

29 27 29 SINK

30 26 30 SINK

31 24 31 SINK

32 23 32 SINK

COMP REF (1.A3)

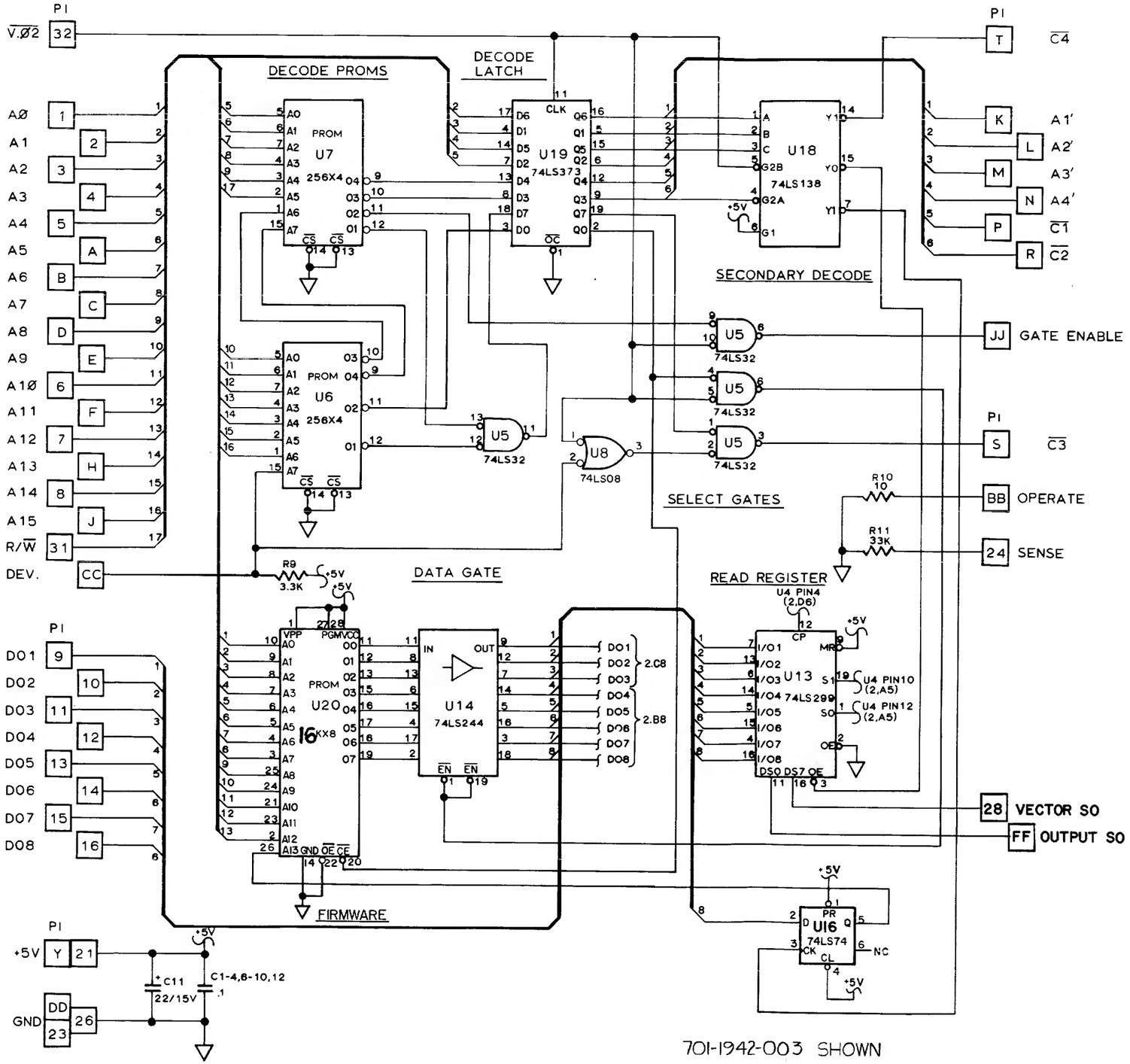
OUTPUT SO (1.C8)

CP (1.B8)

S/L (1.B8)

| APPROVALS: | | UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES. | | DATA I/O | |
|---|---|---|---|---|---|
| DSN. ENGR | DATE | TOLERANCES, UNLESS OTHERWISE SPECIFIED XX . XXX ANGULAR | | TITLE SCHEMATIC DIAGRAM, T/RISE COMPARATOR | |
| MFG. ENGR | DATE 7/16/82 | | | | |
| QUAL ASSUR | DATE 12/15/82 | DRAWN BY | DATE 2/25/82 | SIZE D | CODE IDENT NO 54193 |
| ENGR. MGR. | DATE 11/11/87 | CHECKED BY: | DATE 4/4/12 | DRAWING NO 30-701-1941 | |
| | | | | SCALE | SHEET 2OF2 |

| REVISIONS | | | | | |
|---|---|---|---|---|---|
| LTR | DESCRIPTION | DR | CHK | APPR'D | DATE |
| A | RELEASE | JK | | GR | 6-30 |
| B | ECN 4788 | JK | | GR | 3/83 |
| C | ECN #4785 3-83 | SH | | GR | 3/83 |
| D | ECN 4866, CMA 0039 | CL | | GR | 8/83 |




701-1942-003 SHOWN

NOTES: UNLESS OTHERWISE SPECIFIED

1. ALL RESISTORS ARE IN OHMS, 1/4W, 5%.
2. ALL CAPACITORS ARE IN MICROFARADS, 50V.
3. LAST REFERENCE DESIGNATOR USED: U20, C13, R11, Y1, J1
4. POWER AND GROUND

| REF DES | GND | +5V |
|---|---|---|
| 4,5,6,15-17 | 7 | 14 |
| U1,U2,3,6,7,10,18 | 6 | 16 |
| U9 11,12,13,14,19 | 10 | 20 |

| APPROVALS: | | UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES | DATA I/O | |
|---|---|---|---|---|
| DSN ENGR. GR | DATE 7/12/82 | TOLERANCES, UNLESS OTHERWISE SPECIFIED XX ± XXX ± ANGULAR | TITLE SCHEMATIC DIAGRAM, CONTROLLER/MEMORY | |
| MFG ENGR. [illegible] | DATE 7/12/82 | | | |
| QUAL ASSUR. [illegible] | DATE JUL 12 '82 | DRAWN BY: JAEHEE KIM | DATE 1/82 | SIZE D / CODE INDENT. NO. 54193 / DRAWING NO 30-701-1942 |
| ENGR. MGR G. [illegible] | DATE 7/12/82 | CHECKED BY: [illegible] | DATE 7/12 | SCALE — / SHEET 1OF2 |

| REVISIONS | | | | | |
|---|---|---|---|---|---|
| LTR | DESCRIPTION | DR | CHK | APPR'D | DATE |
| A | SEE SHEET ONE | JK | 4/5/83 | GP | 6-30 |
| B | " | JK | 7/13/83 | GP | 3/83 |
| C | " 3-83 | SH | now 7/21 | GP | 3/83 |
| D | " | CL | | | |

MASTER OSC.

Y1 4MHZ

U15 74LS04

C5 470PF 100V

C13 100PF 200V

R3 1K

R4 1K

U15 74LS04

U4 74LS04

U13 PIN12 (1,B5)

CONTROL COUNTERS

+5V

U2 74LS161

U3 74LS161

VECTOR COUNTER

U1 CD4020B

R1 3.3K

R2 3.3K

U4 74LS04

P1

CP X

SYNC REG.

C6 U

U16 74LS74

NC

CP AND CE TEST PROM

U10 PROM 256X4 TS

U9 74LS273

CONTROL REG.

R5 R6 R7 R6 2.2K (4PLCS)

U17 74LS05

NN CP1

34 CP2

V CE1

19 CE2

U12 74LS273

MODE REGISTER

D01 D02 D03 D04 D05 D06 D07 D08

I.B5

CONTROLLER

U11 PAL 16L8

IN OUT

U13 PIN1 (1,B4)

U13 PIN19 (1,B4)

RUN

TEST

CE GATE

STATUS GATE

U8 74LS08

U8 74LS06

U15 74LS04

CP AND CE BUFFERS

27 CP1 EN

EE CP2 EN

20 STATUS

HH S/L

W RICR

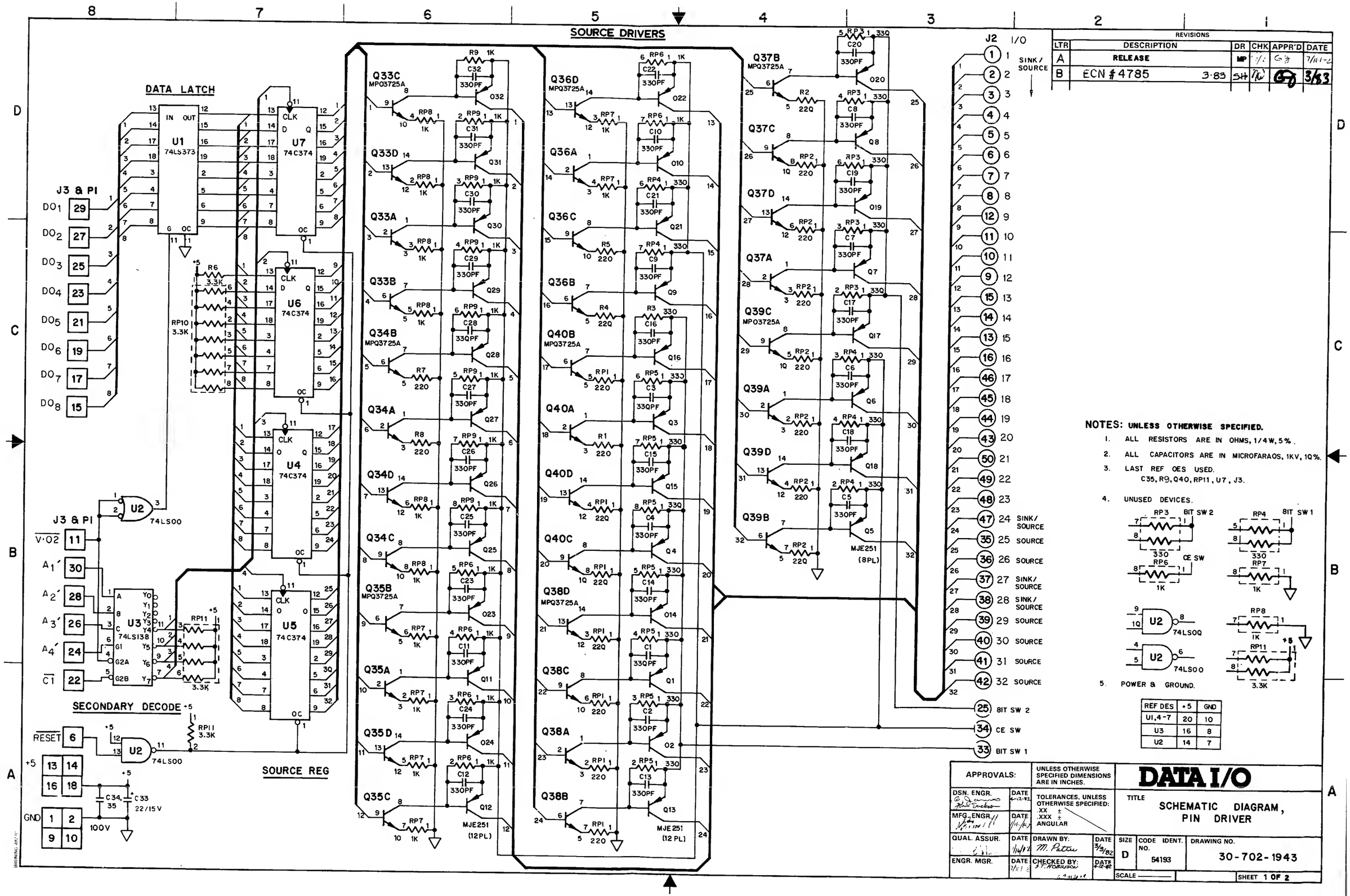
29 IRC

UNUSED GATES:

U15 74LS04

U17 74LS05

| APPROVALS: | | UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES | DATA I/O | |
|---|---|---|---|---|
| DSN ENGR | DATE | TOLERANCES, UNLESS OTHERWISE SPECIFIED XX ± XXX ± ANGULAR | TITLE SCHEMATIC DIAGRAM, CONTROLLER/MEMORY | |
| MFG ENGR. | DATE | | | |
| QUAL. ASSUR. | DATE | DRAWN BY JAEHEE KIM | DATE 1/82 | SIZE D |
| ENGR. MGR. | DATE | CHECKED BY | DATE | CODE INDENT NO. 54193 |
| | | | | DRAWING NO 30-701-1942 |
| | | | | SCALE — SHEET 2OF2 |




| REVISIONS | | | | | |
|---|---|---|---|---|---|
| LTR | DESCRIPTION | DR | CHK | APPR'D | DATE |
| A | RELEASE | MP | | | 7/11/82 |
| B | ECN #4785 3-83 | SH | | | 3/83 |

**NOTES:** UNLESS OTHERWISE SPECIFIED.

1. ALL RESISTORS ARE IN OHMS, 1/4W, 5%.
2. ALL CAPACITORS ARE IN MICROFARADS, 1KV, 10%.
3. LAST REF DES USED.
   C35, R9, Q40, RP11, U7, J3.
4. UNUSED DEVICES.

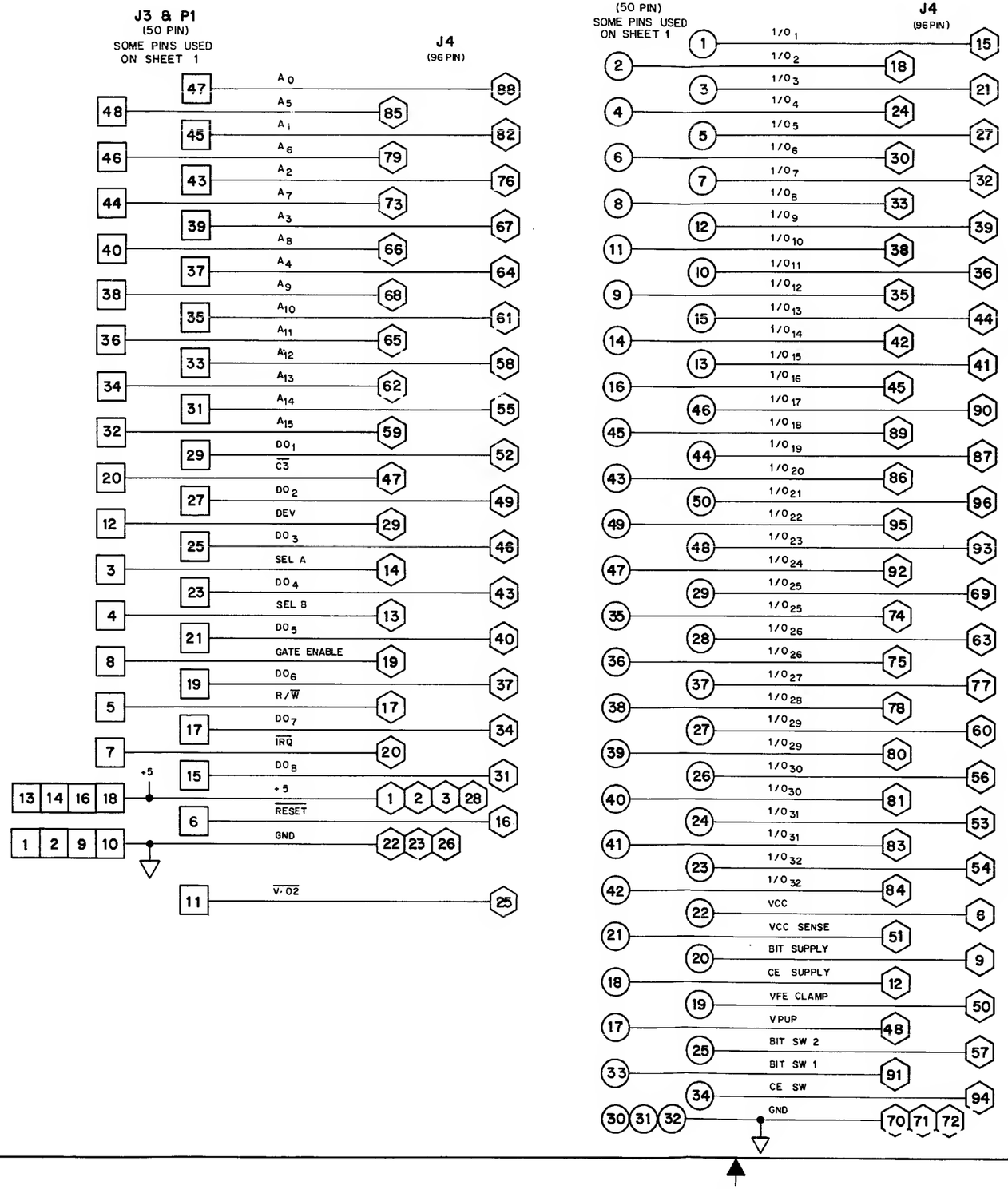



5. POWER & GROUND.

| REF DES | +5 | GND |
|---|---|---|
| U1,4-7 | 20 | 10 |
| U3 | 16 | 8 |
| U2 | 14 | 7 |

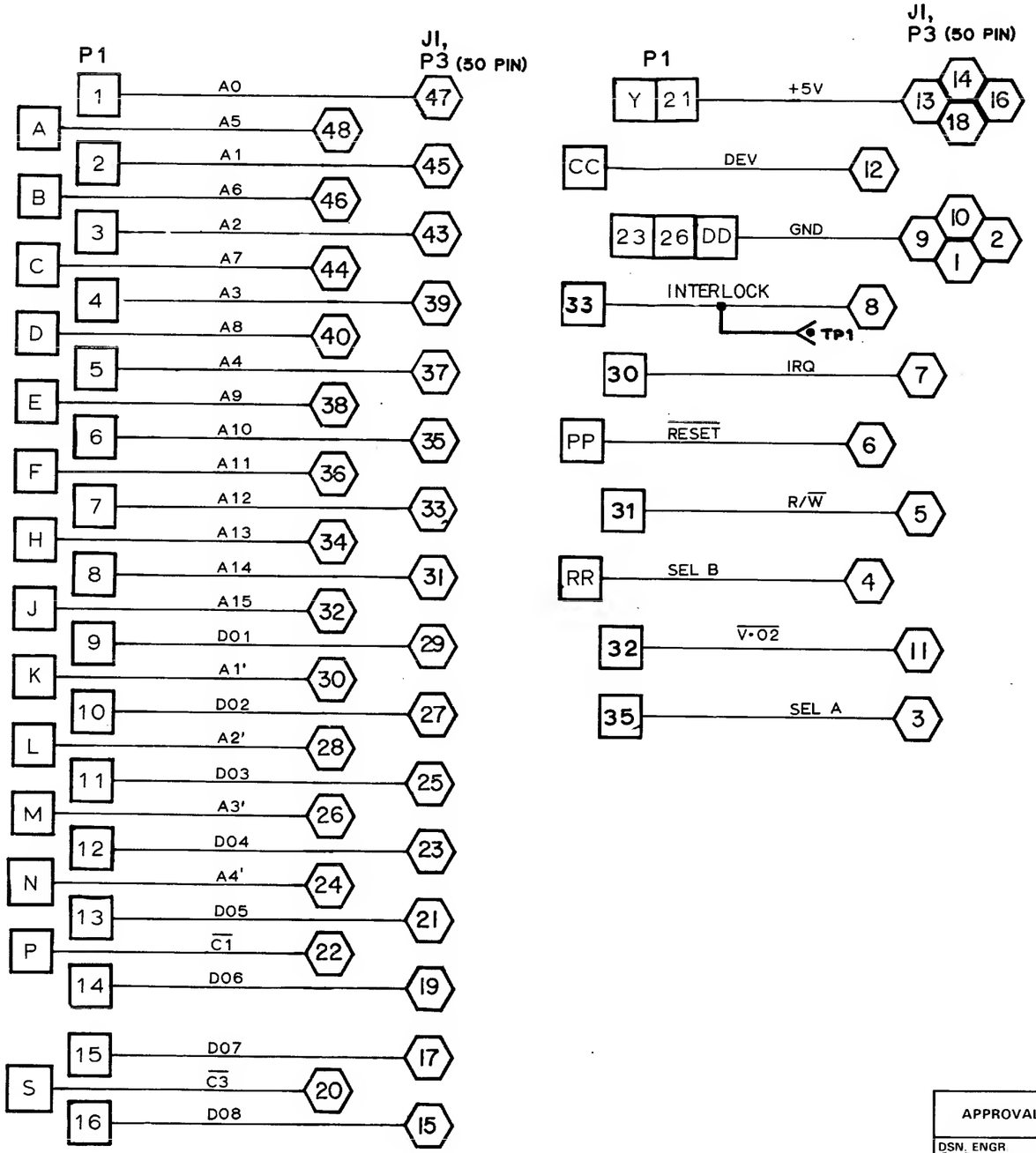
| APPROVALS: | | UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES. | | DATA I/O | | | |
|---|---|---|---|---|---|---|---|
| DSN. ENGR. | DATE 4-12-82 | TOLERANCES, UNLESS OTHERWISE SPECIFIED: XX ± .XXX ± ANGULAR | | TITLE SCHEMATIC DIAGRAM, PIN DRIVER | | | |
| MFG. ENGR. | DATE 4/14/82 | | | | | | |
| QUAL. ASSUR. | DATE 4/14/82 | DRAWN BY: M. Patten | DATE 3/3/82 | SIZE D | CODE IDENT. NO. 54193 | DRAWING NO. 30-702-1943 | |
| ENGR. MGR. | DATE 4-12-82 | CHECKED BY: J.T. ROBINSON | DATE 4-12-82 | SCALE | | SHEET 1 OF 2 | |

| REVISIONS | | | | | |
|---|---|---|---|---|---|
| LTR | DESCRIPTION | DR | CHK | APPR'D | DATE |
| A | SEE SHT 1 | | JZ | | |
| B | 〃 3-83 | SH | MW | GG | 3/83 |




| APPROVALS: | | UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES. | | DATA I/O | |
|---|---|---|---|---|---|
| DSN. ENGR. G. Jamison / John Trucker | DATE 4-12-82 | TOLERANCES, UNLESS OTHERWISE SPECIFIED: .XX ± .XXX ± ANGULAR | | TITLE SCHEMATIC DIAGRAM, PIN DRIVER | |
| MFG. ENGR. | DATE | | | | |
| QUAL. ASSUR. | DATE | DRAWN BY: M. Patru | DATE 3/3/82 | SIZE D | CODE IDENT. NO. 54193 |
| ENGR. MGR. | DATE | CHECKED BY: S.T. ROBILLARD | DATE 4/12/82 | DRAWING NO. 30-702-1943 | |
| | | | | SCALE — | SHEET 2 OF 2 |

| REVISIONS | | | | | |
|---|---|---|---|---|---|
| LTR | DESCRIPTION | DR | CHK | APPR'D | DATE |
| A | RELEASE | R.B. | 3/14 | | 11/1/82 |
| B | ECN #4685 | CL | | | |
| C | ECN #4785 | SH | 7/16 | CS | 3/83 |




## NOTES:

1. LAST REFERENCE DESIGNATOR USED: P3, JI, **TP1**

| APPROVALS: | | UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES | | DATA I/O | |
|---|---|---|---|---|---|
| DSN. ENGR John Erickson | DATE 6-9-82 | TOLERANCES, UNLESS OTHERWISE SPECIFIED: .XX ± .XXX ± ANGULAR | | TITLE SCHEMATIC DIAGRAM, EXTENDER BOARD 50/72 | |
| MFG. ENGR | DATE 7/12/82 | | | | |
| QUAL. ASSUR | DATE JUL 12 '82 | DRAWN BY R. BENN | DATE 12-17-81 | SIZE C | CODE IDENT NO 54193 |
| ENGR MGR | DATE 7/12/82 | CHECKED BY J. ROBINSON | DATE | DRAWING NO 30-701-1944 | |
| | | | | SCALE — | SHEET 1 OF 1 |

# U.S. and Canada Sales Offices and Representatives
# Direct Sales and Service Offices

**HAWAII, OREGON, WASHINGTON**

**Data I/O Corporation**
10525 Willows Road N.E.
C-46
Redmond, WA 98052
(206) 881-6444
Telex 15-2167
*(Sales & Service)*

**NORTHERN CALIFORNIA**

**Data I/O Corporation**
473 Sapena Court
Suite 4
Santa Clara, CA 95050
(408) 727-0641
Telex 352054
*(Sales, Applications & Service)*

**WESTERN REGION**

**Data I/O Corporation**
2770 South Harbor Blvd.
Suite K
Santa Ana, CA 92704
(714) 662-1182
Telex 910-595-1562
*(Sales & Service)*

**EASTERN REGION ADMINISTRATIVE OFFICES**

**MAINE, EASTERN MASSACHUSETTS, NEW HAMPSHIRE, VERMONT**

**Data I/O Corporation**
Nashua Road
Route 101A
Amherst, NH 03031
(603) 889-8511 *(Sales)*
(603) 889-8513 *(Applications & Service)*
Telex 943431

**SOUTHERN REGION ADMINISTRATIVE OFFICE**

**Data I/O Corporation**
1810 N. Glenville Drive
Suite 108
Richardson, TX 75081
(214) 235-0044
Telex 792474
*(Sales & Applications)*

## Representatives

*(Sales Only)*

**ALASKA**

**Microprocessor Technology**
3537 East 65th Avenue
Anchorage, AK 99507
(907) 344-8080
Telex (090) 25-462

**OREGON, EASTERN WASHINGTON**

**Northwest Test and Measurement**
8196 Southwest Hall
Suite 217
Beaverton, OR 97005
(503) 646-9966

**ARKANSAS, LOUISIANA, OKLAHOMA, TEXAS**

**TESTECH, Inc.**
1000 E. Campbell Road
Suite 108
Richardson, TX 75081
(214) 644-5010

9219 Katy Freeway
Suite 124
Houston, TX 77024
(713) 467-2192

2525 Wallingwood
Suite 804
Austin, TX 78746
(512) 327-7033

**MICHIGAN, EASTERN KENTUCKY, OHIO, WEST VIRGINIA, WESTERN PENNSYLVANIA**

**Electro Sales Associates**
1635 Mardon Drive
Dayton, OH 45432
(513) 426-5551

29200 Vassar Road
Suite 505
Livonia, MI 48152
(313) 474-7320

9816 Portage Road
Portage, MI 49002
(616) 323-2416

851 East 222nd Street
Cleveland, OH 44123
(216) 261-5440

3740 Mount Royal Blvd.
Allison Park, PA 15101
(412) 487-3801

**UPSTATE NEW YORK**

**Martin P. Andraws**
523 East Genesee Street
P.O. Box 443
Fayetteville, NY 13066
(315) 637-5291

**METROPOLITAN NEW YORK, LONG ISLAND, NORTHERN NEW JERSEY**

**RTI**
300 Northern Blvd.
Great Neck, NY 11021
(516) 829-3770

**KANSAS, MISSOURI, NEBRASKA, SOUTHERN ILLINOIS**

**Midtac Associates, Inc.**
7702 Mize Road
DeSoto, KS 66018
(913) 441-6565
Telex 910-743-4129

8460 Lindbergh North
Suite 11
Florissant, MO 63031
(314) 837-5200

**ALABAMA, FLORIDA, GEORGIA, MISSISSIPPI, NORTH CAROLINA, SOUTH CAROLINA, TENNESSEE**

**Pen Tech Associates**
1398 Semoran Blvd.
Suite 106
Casselberry, FL 32707
(305) 645-3444

201 S.E. 15th Terrace
Suite K
Deerfield Beach, FL 33441
(305) 421-4989

Cherokee Center
627 Cherokee Street
Suite 21
Marietta, GA 30060
(404) 424-1931

Holiday Office Center
3322 Memorial Pkwy. S.W.
Suite 36
Huntsville, AL 35801
(205) 881-9298

3709 Alliance Drive
Suite C
Greensboro, NC 27407
(919) 852-6000

**INDIANA, IOWA, WESTERN KENTUCKY, MINNESOTA, NORTHERN ILLINOIS, NORTH DAKOTA, SOUTH DAKOTA, WISCONSIN**

**Torkelson Associates**
4940 Viking Drive
Minneapolis, MN 55435
(612) 835-2414

108 Wilmot Road
Suite 110
Dearfield, IL 60015
(312) 945-8700

2346 S. Lynhurst
Room B101
Indianapolis, IN 46241
(317) 244-7867

2840 N. Brookfield Road
Brookfield, WI 53005
(414) 784-7736

**ARIZONA, COLORADO, IDAHO, MONTANA, NEW MEXICO, NEVADA, UTAH, WYOMING**

**Zeus Electronics, Inc.**
1428 E. Pierson Street
Phoenix, AZ 85014
(602) 263-6022
Telex 910-951-1362

310 Val Verde S.E.
Albuquerque, NM 87108
(505) 842-6633

3333 Quebec Street
Suite 2950
Denver, CO 80207
(303) 321-4246

1849 W. North Temple
Suite B105
Salt Lake City, UT 84116
(801) 534-0500

**SOUTHERN NEW JERSEY, DELAWARE, MARYLAND, EASTERN PENNSYLVANIA, VIRGINIA, WASHINGTON D.C.**

**SCI-REP, Inc.**
304 Cooper Center
Pennsauken, NJ 08109
(609) 662-5222

9512A Lee Highway
Fairfax, VA 22031
(703) 385-0600

**CONNECTICUT, RHODE ISLAND, WESTERN MASSACHUSETTS**

**Pat Jenks Associatas**
52 Washington Avenue
North Haven, CT 06473
(203) 239-6201

114 Cummings Park
Woburn, MA 01801
(617) 938-0488

**CANADA**

**Allan Crawford Associates, Ltd.**
6503 Northam Drive
Mississauga, Ontario L4V 1J2
(416) 678-1500
Telex 06 968769

881 Lady Ellen Place
Ottawa, Ontario K1Z 5L3
(613) 722-7582
Telex 053 3600

7018 Cote De Liesse
St. Laurent, P.Q. H4T 1E7
(514) 731-3564
Telex 05-824944

1935-30th Avenue N.E.
Calgary, Alberta T2E 6Z5
(403) 230-1341
Telex 03 821186

3795 William Street
Burnaby, B.C. V5C 3H3
(604) 294-1326
Telex 04 54247

192 Joseph Zatzman Drive
Dartmouth, Nova Scotia B3B 1N4
(902) 463-9360
Telex 019 31604

15043A 118th Avenue
Edmonton, Alberta T5V 1H9
(403) 451-4893

Addresses on this list subject to change without notice.

0321/183

# International Sales Offices and Representatives

## Europe

**Date I/O Europe**
Vondelstraat 50-52
NL-1054 GE Amsterdam
The Netherlands
(20)186855
Telex 16616 DATIO NL

**Deta I/O Germany GmbH**
Bahnhofstrasse 3
D-6453 Seligenstadt
West Germany
(6182)3088/89
Telex 4184962 DATA D

**AUSTRIA**

**Ing. Ernst Steiner**
Hummelgasse 14
A-1130 Wien
Austria
(222)827474
Telex 135026 ES A

**BELGIUM**

**Simac Electronics**
Rue du Progres 52
BOITE 3
B-1000 Brussels
Belgium
(2)2192451-3
Telex 23662 SIMEIP B

**DENMARK**

**ITT Komponent A/S**
Fabriksparken 31
DK-2600 Glostrup
Denmark
(2)451822
Telex 33355 itt dk

**FINLAND**

**Hevulinna-Instrumenterium Oy**
P.O. Box 357
SF-00101 Helsinki 10
Finland
(0)799711
Telex 124426 havul sf

**FRANCE**

**M.B. Electronique**
606, rue Fourny
Zac de Buc
P.O. Box 31
F-78530, Buc, France
(3)9568131
Telex MB 695414 F

**GERMANY**

**Instrumetic Electronics GmbH**
Am Kirchenholzel 14
D-8032 Grafelfing (near Munich)
West Germany
(89)852063
Telex 524298 INSTR D

**GREECE**

**Mr. Elies Memmees**
65 Meg. Alexandrou & Athinas Street
P.O. Box 181
Korydallows
Piraeus
(1)4967815/or 818
Telex 213835 LH GR

**ISRAEL**

**R.D.T. Electronics Engineering, Ltd.**
46 Sokolov Street
P.O. Box 75
Ramat Hasharon
Israel
(3)483211
Telex 32143 RDT IL

**ITALY**

**Sistrel SPA**
Via Giuseppe Armellini 39
I-00143 Rome
Italy
(6)5915551
Telex 680356 SISTREL i

**Sistrel SPA**
Via P. da Volpedo 59
I-20092 Cinisello Balsamo (Mi)
Italy
(2)6120129 or 6181893
Telex 334643 SISTREL i

**NETHERLANDS**

**Simac Electronics**
Veenstraat 20
NL-5503HR Veldhoven
The Netherlands
(40)533725
Telex 51037 SIMAC NL

**NORWAY**

**Teleinstrument A/S**
P.O. Box 134
N-1371 Asker
Norway
(2)789460
Telex 72919 TELIN N

**PORTUGAL**

**Decada**
Equipamentos de Electronica, Lda.
Rua Pedro Nunes 47-C
P-1000 Lisboa or P-1003 Lisboa Codex
Portugal
(19)574984
Telex 18469 NONIO P

**SPAIN**

**Instrumatic**
Alameda Principal 26
Apartado 151
Malaga 5
Spain
(52)213199/213898
Telex 77131 Hafn e

**Instrumetic**
Juan Hurtado de Mendoza 9
Madrid-16
Spain
(1)2502577/2507278
Telex 46277 Itic e

**SWEDEN**

**Mecrotek AB**
Vallingbyvagen 212, Box 43
S-162 11 Vallingbyvagen, Sweden
(8)870190
Telex 12543 MATEK S

**SWITZERLAND**

**Instrumatic SA**
5-7, rue du Clos
CH-1207 Geneve
Switzerland
(22)360830
Telex 28667 FLUJA CH

**Instrumetic AG**
Weingartenstrasse 9
CH-8803 Ruschlikon
Switzerland
(1)7241410
Telex 56605 INST CH

**TURKEY**

**Dete I/O Europe**
Vondelstraat 50-52
NL-1054 GE Amsterdam
The Netherlands
(20)186855
Telex 16616 DATIO NL

**UNITED KINGDOM**

**Microsystem Services**
P.O. Box 37
Lincoln Road
Cressex Industrial Estate
High Wycombe
Bucks, HP12 3XJ, England
(494)41661
Telex 837187 MICSYS G

## International

**Deta I/O International**
10525 Willows Road N.E.
C-46
Redmond, WA 98052
U.S.A.
(206) 881-6444
Telex 15-2167

**AUSTRALIA**

**CORPORATE OFFICE**

**Werburton O'Donnell Ltd.**
372 Eastern Valley Road
P.O. Box 182
Chatswood, N.S.W., Australia 2067
4073261
Telex Warfran AA21299

**SOUTH AUSTRALIA**

**ADELAIDE, AUSTRALIA**

**Werburton Franki (Adelaide) Pty. Ltd.**
322 Grange Road
Kidman Park
South Australia 5025 Australia
3567333
Telex Warfran AA82579

**BRISBANE, QUEENSLAND**

**Werburton Frenki (Brisbene) Pty. Ltd.**
13 Chester Street
Fortitude Valley
Queensland 4006 Australia
527255
Telex Warfran AA41052

**MELBOURNE, VICTORIA**

**Werburton Frenki (Melbourne) Pty. Ltd.**
220 Park Street
South Melbourne
Victoria 3205 Australia
699 4999
Telex Warfran AA31370

**PERTH, WESTERN AUSTRALIA**

**Werburton Frenki (Perth)**
98-102 Belgravia Street
Belmont, 6104
Western Australia
65 7000
Telex Warfran AA92908

**SYDNEY, NEW SOUTH WALES**

**Werburton Frenki Sydney**
199 Parramatta Road
Auburn, N.S.W. 2144
Australia
648 1711
Telex Warfran AA22265

**TASMANIA**

**Associeted Agencies Pty., Ltd.**
P.O. Box 252
43 Albert Road
Moonah, Tasmania 7009
23 1841
Telex ASSAG AA58206

**BRAZIL**

**Cosele**
Instrumentos Electronicos Ltda.
Rue da Consolacao, 867
3.° And. - Conj. 32
01301 - Sao Paulo - SP
Brasil
255-1733 or 256-7421 or 231-5548
Telex (011) 30869 CSEL-BR

**HONG KONG**

**Cummings Engineering Co., Ltd.**
90 Sung Wong Toi Road, I/F,
To Kwa Wan, Kowloon,
Hong Kong
3-636227
Telex 74015 CUMMH HX

**INDIA**

**Transmerketing Privete, Ltd.**
Sterling Center
16/2 Dr. Annie Besant Rd.
Bombay 400-18
India
022 391874
Telex 011-5424

**JAPAN**

**Deta I/O Jepen Compeny, Ltd.**
Sankei Building
1-8, Sarugaku-cho 2-Chome
Chiyoda-Ku, Tokyo 101
Japan
(03) 295-2656
Telex 2225391 DATAIO J

**KOREA**

**Elcom System, Inc.**
Yeongdong Box 50
Seoul, Korea
555-5222 or 557-3836
Telex ADUCEL K25227

**MALAYSIA**

**Cedco Technicel Services**
67A, Jalan Perisai
Taman Sri Tebrau
Johore Bahru, Malaysia
Ceble - CADCOKIT - JOHOR BAHRU

**MEXICO**

**Christensen, S.A.**
Guillermo Prieto 76-304
Col. San Rafael
Delegecion Cuahutemoc
06470-Mexico, D.F.
546-25-95
546-29-55
Telex 017-75612 Mycome.

**NEW ZEALAND**

**Wellington-New Zeelend:**
**Werburton Frenki, Ltd.**
42-43 Oxford Terrace
Lower Hutt, New Zealand
693-016
Telex Warfran NZ 3824

**Werburton Franki, Ltd.**
P.O. Box 9301; Newmarket
142 Broadway
Auckland, New Zealand
504-458
Telex Werfran NZ 3824

**PHILIPPINES**

**Cummings Engineering Co., Ltd.**
90 Sung Wong Toi Road, 1/F,
To Kwa Wan, Kowloon
Hong Kong
3-636227
Telex 74015 CUMMH HX

**SINGAPORE**

**GEA Technology PTE., Ltd.**
Units 1003 to 1008, Block 3, 10th Floor
PSA Multi-Storey Complex
Pasir Panjang Road
Singapore 0511
2729412
Telex RS 37162 Answerback Gaasin

**SOUTH AFRICA**

**Electronic Building Elements**
**(PTY) Ltd.**
P.O. Box 4609
Pine Square, 18th Street
Hezelwood
Pretorie, South Africa 0001
46-9221/7
Telex 3 0181 SA
Telegrams Elbilem

**SOUTH & CENTRAL AMERICA**

**Dexter Computer Products, Inc.**
3465 Torrance Blvd.
Suite N
Torrence, CA 90503
213-542-7333
Telex 182360 DXTR TRNC

**TAIWAN**

**COMPAC Microelectronics, Inc.**
11th Fl., Walsin Bldg.
#219, Chung-Hsiao E. Road, Sec. 4
P.O. Box 53-479
Taipei, Taiwan R.O.C.
7529911
Telex 24368 COMTWN

**THAILAND**

**Dynamic Supply Engineering R.O.P.**
12 Soi Pasara 1, Sukhumvit 63
Bangkok-11, Thailand
Tel. - 3914434, 39228532
Telex 82455 DYNASUP TH

# ATTENTION
# MANUAL CHANGE INFORMATION
# INSTRUCTIONS

At Data I/O, we continually strive to improve our equipment by adding new improvements to our hardware and software as soon as they are developed and tested.

The attached Documentation Change information reflects an update to this equipment. Please file the Document Change Summary and Document Change Revised Pages Record forms in front of the Title Page in your manual. The format of the Document Change information you receive will vary. Follow the instructions below for the correct procedure.

1. Document Addendum: File Document Addendums behind the Document Change Revised Pages Record.
2. Revised Pages: Replace the existing pages in the manual with the revised pages.
3. Complete Revision: Remove all existing pages, including Document Addendums, and replace them with the newly issued revision.

If you have any questions regarding these changes, please contact:

Ms. Jan Simmons
Technical Publications Supervisor
Data I/O Corporation
10525 Willows Road NE/C-46
Redmond, WA 98052
(206) 881-6444
Telex 15-2167

# DOCUMENT CHANGE SUMMARY

At Data I/O, we continually strive to improve our instruments as soon as they are developed. Sometimes, because of printing and shipping requirements, we can't include these changes immediately into printed manuals. Consequently, your manual may contain new change information as summarized below.

File the Documentation Change Summary and the Document Change Record pages in the beginning of your manual, in front of the Title Page. File any Document Addendum pages received with changes immediately after the other two Document Change pages. If you have received revised pages, remove the previous page version, and replace it with the latest page revision. If you have received a complete revision, remove all previously filed Document Addendums and replace the existing manual pages with the revised manual pages. Whenever you receive a manual update, replace the existing Document Change pages with the most recent set of Document Change pages.

| REV | DESCRIPTION OF CHANGE | DATE |
|---|---|---|
| C | Incorporate ECN# 4958 | 10/83 |
| D | Incorporate ECN# 4968 | 10/83 |

0547/1282

# DOCUMENT CHANGE REVISED PAGES RECORD

| REVISED | | | | ADDED | | DELETED | |
|---|---|---|---|---|---|---|---|
| PAGE NUMBER | REV & DATE | PAGE NUMBER | REV & DATE | PAGE NUMBER | REV & DATE | PAGE NUMBER | REV & DATE |
| Title page | C 10/83 | | | | | | |
| Title page | D 10/83 | | | | | | |

0546/1282

# Warranty 90 Days

Data I/O warrants that all equipment sold pursuant to any resultant agreement shall be free from defects in material or workmanship at the time of delivery. Such warranty shall extend for 90 days. Buyer must provide notice to Data I/O within this prescribed warranty period of any defect if the defect is not the result of improper usage, service, maintenance or installation and equipment has not been otherwise damaged or modified after delivery. Data I/O shall either replace or repair the defective part or parts of equipment or replace the equipment or refund the purchase price at Data I/O's option after return of such equipment by buyer to Data I/O. Shipment to and from Data I/O's facility shall be borne on account of buyer.

(a) Consequential Damages — Data I/O shall not be liable for any incidental or consequential damages incurred as a result of any defect in any equipment sold hereunder, and Data I/O's liability is specifically limited to its obligation described herein to repair or replace a defective part or parts covered by this warranty.

(b) Exclusive Warranty — The warranty set forth herein is the only warranty, oral or written, made by Data I/O and is in lieu of and replaces all other warranties, expressed or implied, including the WARRANTY OF MERCHANTABILITY AND THE WARRANTY OF FITNESS FOR A PARTICULAR PURPOSE.

0299/6K/183

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

## BUSINESS REPLY CARD

FIRST CLASS PERMIT NO. 41 REDMOND, WA 98052

POSTAGE WILL BE PAID BY ADDRESSEE

**DATA I/O**
Reader Comments
Applications Manager
10525 Willows Road N.E.
C-46
Redmond, WA 98052

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY CARD

FIRST CLASS PERMIT NO. 41 REDMOND, WA 98052

POSTAGE WILL BE PAID BY ADDRESSEE

**DATA I/O**
Warranty/Registration
Service Manager
10525 Willows Road N.E.
C-46
Redmond, WA 98052

# Reader Comments

The manual's completeness, accuracy, organization, usability, and reliability: _____

_____

Did you find errors in this manual? _____

How can this manual be improved? _____

_____

_____

Additional comments: _____

_____

Name _____ Title _____

Department _____ M/S _____

Company _____

Address _____

City _____ State _____ Zip _____

Phone ( ) _____

Manual or Part No. _____

# Warranty/Registration

*Please return this card to Data I/O within five days*

Did the packaging of this equipment exhibit any outward signs of physical damage? ☐ YES ☐ NO

Did this equipment arrive intact, without loose parts or cable damage? ☐ YES ☐ NO

Did the equipment operate on power-up? ☐ YES ☐ NO

Did you attain adequate system performance? ☐ YES ☐ NO

Were any electrical adjustments required? ☐ YES ☐ NO

If you required assistance, was a local Data I/O representative contacted? ☐ YES ☐ NO

Comments: _____

_____

Name _____ Title _____

Department _____ M/S _____

Company _____

Address _____

City _____ State _____ Zip _____

Phone ( ) _____

Model or Description _____ Serial or Part No. _____

Date Received _____